COMP20005 Workshop Week 10

1	Structures: Ex 8.1
2	Polygons Revisited (Ex. 8.2 & 8.3)
3	Assignments: Ass1 Review, Ass2 Q&A
4	Lab: Work on ass2, and/or: • fully implement ex 8.2-8.4 using array-based poly.c and data file polys.txt, both supplied in github • explore exercise 8.9 in github

Structures Revisited

Ex 8.1 (simplified)

Supposing:

`NAMELEN is 40`

2) Draw a diagram for `name`.

2) Write input statement for `name`

3) How many bytes used:

- `name`
- `date`

```
typedef char namestr[NAMELEN+1];

typedef struct {
   namestr given, others, family;
} name_t;

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

name_t name;
date_t date;
```

Structures Revisited

Ex 8.1 (simplified)

Supposing:

`NAMELEN` and `STUDSMAX` are `40` and `50000`

1) Draw a diagram for `stud`.

2) How many bytes used:

- `stud`
- `unimelb`

```
typedef struct {
    name_t name;
    int id;
    date_t dob;
    // int ns;
    // subject_t subs[SUBSMAX];
} student_t;

student_t stud;
student_t unimelb[STUDSMAX];
```

pointers to struct

```
typedef struct {
    name_t name;
    int id;
    date_t dob;
    // int ns;
    // subject_t subs[SUBSMAX];
} student_t;

student_t stud;
student_t *ps;
ps = &stud;
```

member `id` of `stud` can be written as `stud.id`. It can also be written as `(*ps).id` and `ps->id`

`ps->` is a shorthand for `(*ps).`

pointers to struct

```
typedef struct {
    name_t name;
    int id;
    date_t dob;
    // int ns;
    // subject_t subs[SUBSMAX];
} student_t;

student_t stud;
student_t *ps;
ps = &stud;

scanf("%s%s%s%d%d%d%d", stud.name.given,...,
        &stud.id, &stud.dob.dd, &stud.dob.mm,...);

scanf("%s%s%s%d%d%d%d", ps->name.given,...,
        &ps->id, &ps->dob.dd, &ps->dob.mm,...);
```

Structures revisited

Write a function header and function call for inputting one student record into variable stud.

How many bytes are passed:

- (at the beginning) from the caller to the function, and
- (at the end) from the function to the caller.

write function to read a student

```
typedef struct {
    name_t name;
    int id;
    date_t dob;
} student_t;


?? read_stud( ??? ) {

    scanf("%s%s%s %d %d/%d/%d",




}
```

Structures: function for input, version 1 (bad)

```
student_t 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;
}
```

How many bytes copied with the function call?

```
student_t stud;
stud= read_stud();
```

Structures: function for input, version 2

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

How many bytes copied with the function call?

```
student_t stud;
readtime(&stud);
```

Structures: important rule

Don't use a `struct` for a function argument,
or as returned value from functions

use a `pointer to struct` instead.

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.*

a) Build a data file `polys.txt` *with content:*

```
3  0 0  3 0  0 4
4  5 0  6 0  6 1  5 1
```

which represent a triangle and a square.

b) Write a program that includes the following functions that

(i) reads a poly from `stdin`

(ii) returns the length of the perimeter of a polygon (ex 8.3).

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

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

Test these functions using data from `polys.txt`*.*

ass1 review: a sample solution

assignment 2: new in rubric

- avoidance of structs (eg, using multiple 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 old, 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;
- runtime segmentation fault on any other test with no output generated, -2.0;

assignment 2: input data

```
(fixed)Code, Station, Year,  Month, Rainfall, Validated?
IDCJAC0001, 086039, 2000,    01,    28.2,    Y
IDCJAC0001, 086039, 2000,    02,    34.5,    Y
IDCJAC0001, 086039, 2000,    03,    22.5,    N
IDCJAC0001, 086039, 2000,    05,    96.3,    Y
```

```
scanf("IDCJAC0001,%d,%d,%d,%lf,%c", ... )
```

Notes:

- data lines are in ascending order of (year, month),
- first and last data lines define the time range, but
- there might be some missing lines,
- data for a whole year might be missing.

Stage 1 (tiny.csv)

```
(fixed)Code, Station, Year,  Month, Rainfall, Validated?
IDCJAC0001, 086039, 2000,    03,    22.5,    N
IDCJAC0001, 086039, 2000,    05,    96.3,    Y
IDCJAC0001, 086039, 2003,    06,    28.2,    Y
IDCJAC0001, 086039, 2003,    07,    34.5,    Y
IDCJAC0001, 086039, 2003,    08,    22.5,    N
IDCJAC0001, 086039, 2003,    09,    96.3,    Y
```

./my_ass1 < tiny.csv

Stage 1 ouput

```
(fixed)Code, Station, Year,  Month, Rainfall, Validated?
IDCJAC0001, 086039, 2000,    03,    22.5,    N
IDCJAC0001, 086039, 2000,    05,    96.3,    Y
IDCJAC0001, 086039, 2003,    06,    28.2,    Y
IDCJAC0001, 086039, 2003,    07,    34.5,    Y
IDCJAC0001, 086039, 2003,    08,    22.5,    N
IDCJAC0001, 086039, 2003,    09,    96.3,    Y
```

```
S1, site number 086039, 6 datalines in input
S1, 2000: ...  ...  Mar* ...  May  ...  ...  ...  ...  ...  ...  ...
S1, 2001: ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...
S1, 2002: ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...
S1, 2003: ...  ...  Mar  ...  ...  ...  Jul  Aug* Sep  ...  ...  ...
```

Stage 2 ouput

```
(fixed)Code, Station, Year,  Month, Rainfall, Validated?
IDCJAC0001, 086039, 2000,    03,    22.5,    N
IDCJAC0001, 086039, 2000,    05,    96.3,    Y
IDCJAC0001, 086039, 2003,    06,    28.2,    Y
IDCJAC0001, 086039, 2003,    07,    34.5,    Y
IDCJAC0001, 086039, 2003,    08,    22.5,    N
IDCJAC0001, 086039, 2003,    09,    96.3,    Y
```

```
S1, site number 086039, 6 datalines in input
S2, Jan,  0 values
S2, Feb,  0 values
S2, Mar,  2 values, 2000-2003, mean of  20.4mm
S2, Apr,  0 values
S2, May,  1 values, 2000-2000, mean of  22.3mm
S2, Jun,  0 values
S2, Jul,  1 values, 2003-2003, mean of  34.5mm
S2, Aug,  1 values, 2003-2003, mean of  22.5mm
S2, Sep,  1 values, 2003-2003, mean of  96.3mm
S2, Oct,  0 values
S2, Nov,  0 values
S2, Dec,  0 values
```

Stage 3 ouput

```
(fixed)Code, Station, Year,  Month, Rainfall, Validated?
IDCJAC0001, 086039, 2000,    03,    22.5,    N
IDCJAC0001, 086039, 2000,    05,    96.3,    Y
IDCJAC0001, 086039, 2003,    06,    28.2,    Y
IDCJAC0001, 086039, 2003,    07,    34.5,    Y
IDCJAC0001, 086039, 2003,    08,    22.5,    N
IDCJAC0001, 086039, 2003,    09,    96.3,    Y
```

```
S3, Jan,  0 values
S3, Feb,  0 values
S3, Mar,  2 values, 2000-2003, tau of  1.00
S3, Apr,  0 values
S3, May,  1 values
S3, Jun,  0 values
S3, Jul,  1 values
S3, Aug,  1 values
S3, Sep,  1 values
S3, Oct,  0 values
S3, Nov,  0 values
S3, Dec,  0 values
```