COMP20005 Workshop Week 10

Preparation:

- open `grok`, `jEdit`, and `minGW` (or `Terminal` if yours is a `Mac`)
- open related files for assignment 2

1
2
3

`Discussion 1: Root finding`

- Exercise 9.5 (not in grok)
- Re-examine the cube_root() function on page 77 of the textbook, croot.c. What method does it use? Explore what happens if: (a) very large numbers are provided as input; (b) very small (close to zero) numbers are provided; and (c) CUBE_ITERATIONS is made larger or smaller.

`Discussion 2: struct`,

- Ex. 8.1
- Ex. 8.2-8.4 combined

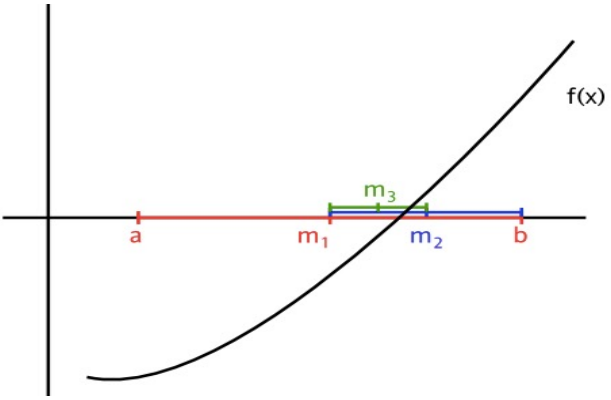
LAB

Assignment 1 review: Questions? Notes for ass2 marking

Assignment 2:

- Watched the assignment movie?
- Q&A
- Working on assignments

Discussion 1: Roof Finding for f(x)=0 using the bisection method



Bracketing Methods `[a mid b]`

Bisection: `mid= (a+b)/2`

False position:

`mid= (af(b)-bf(a))/(f(b)-f(a))`

Transition to next iteration: a=mid or b=mid depending on `bf(mid)<0` or `af(mid)<0`

Methods that build series $x_1, x_2, \ldots, x_n, \ldots$

Secant:

$$x_{k+1} = (f(x_k))x_{k-1} - f(x_{k-1})x_k) \,/\, (f(x_k)\text{-}f(x_{k-1}))$$

Newton-Raphson:

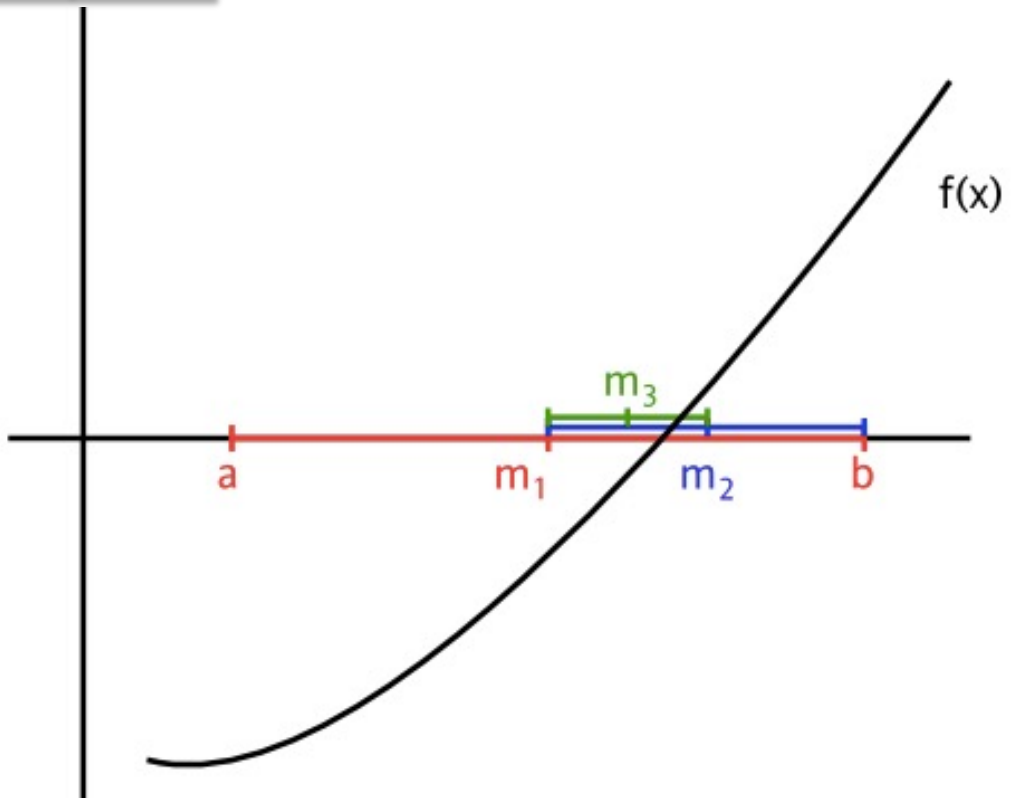
$$x_{i+1} = x_i - f(x_i)/f'(x_i)$$

Root finding: see `numericB.pdf` for methods such as: *bisection,* , *false position*, *fix-point iteration*, *Newton-Raphson*, *secant*

Roof Finding for f(x)=0: bisection method

`mid= (a+b)/2` `(f(a)f(b)<0)`

Transition to next iteration: a=mid or b=mid depending on `af(mid)<0` or `bf(mid)<0`



Ex. 9.5: The square root of z is the of equation $f(x) = x^2 - z$.

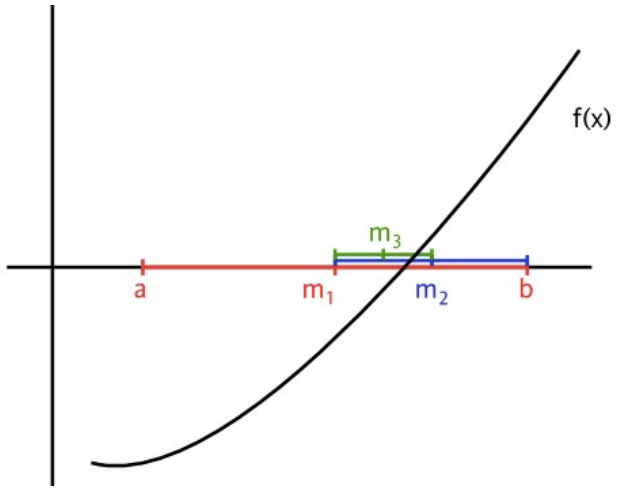
Evaluate the bisection method *by hand* (well, you can use calculators) for z=2, start with a= 1 and b= 3. Stop when the length of the interval is 0.1 or less.

Do it manually and check your answer with me after 3 min

Roof Finding for f(x)=0: bisection method

`mid= (a+b)/2` `(f(a)f(b)<0)`

Transition to next iteration: a=mid or b=mid depending on `af(mid)<0` or `bf(mid)<0`



Ex. 9.5: The square root of *2* is the of equation $f(x) = x^2 - 2$. *Using bisection method* start with a= 1 and b= 3. Stop when the length of the interval is 0.1 or less.

a	b	mid	f(a)	f(b)	f(mid)
1.0000	3.0000	2.0000	-1.0000	7.0000	2.0000
1.0000	2.0000	1.5000	-1.0000	2.0000	0.2500
1.0000	1.5000	1.2500	-1.0000	0.2500	-0.4375
1.2500	1.5000	1.3750	-0.4375	0.2500	-0.1094
1.3750	1.5000	1.4375	-0.1094	0.2500	0.0664
1.3750	1.4375	1.4062	-0.1094	0.0664	-0.0225

```
define NAMESTRLEN 39
#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, status, finalmark;
  date_t enrolled;
} subject_t;

typedef struct {
  fullname_t name;
  date_t dob, datecommenced;
  int id, status, salary;
} staff_t;

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

staff_t alice;
student_t bob;
staff_t allstaff[1000];
student_t allstudents[10000];
```

Discussion 2: struct

Exercise 8.1:

With the declaration on the LHS, how many bytes each of the variables consume?

```
alice       :
bob         :
allstaffs   :
allstudents:
```

Discuss your solutions with friends, then compare with the expected solution (coming shortly).

```
define NAMESTRLEN 39
#define MAXSUBJECTS 8
typedef namestr_t char[NAMELEN+1];       //40

typedef struct {
   namestr_t first, others, family;
} fullname_t;                            //120

typedef struct {
   int dd, mm, yyyy;
} date_t;                                //12

typedef struct {
  int subjectcode, status, finalmark;    //  12
  date_t enrolled;                       //  12
} subject_t;                             //24

typedef struct {
  fullname_t name;                       //  120
  date_t dob, datecommenced;             //   24
  int id, status, salary;                //   12
} staff_t;                               //156

typedef struct {
  int id;                                //    4
  fullname_t name;                       //  120
  date_t dob;                            //   12
  int nsubjects;                         //    4
  subject_t subjects[MAXSUBJECTS];       //  192
} student_t;                             //332

staff_t alice;                    // 156
student_t bob;                    // 332
staff_t allstaff[1000];           // 14,600
student_t allstudents[10000];     // 3,320,000
```

Check 8.1 Solution:

With the declaration on the LHS, how many bytes each of the variables consume?

```
alice       : 156
bob         : 332
allstaffs   : 14,600
allstudents: 332x10^4
```

Check your answer!

struct How to use struct

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

date_t dob={01,01,2001};
date_t yearlater;
date_t *p;

printf("dob= %02d/%02d/%04d\n", dob.dd, dob.mm, dob.yyyy);
yearlater= dob;   // "whole strucrure" assignment is possible
                  //    note that is not applied to arrays
                  //    because arrays are constant pointers

yearlater.yyyy++; // access component using ".", each component
                  //    is just a variable, and can be an array
                  //    or another struct

p= &dob;          // pointer! Bravo!
printf("dob= %02d/%02d/%04d\n", p->dd, p->mm, p->yyyy);
                  //    p->dd is just a shorthand for (*p).dd
```

Review: function for input, version 1 (The Bad)

```
date_t read_date() {
  date_t d;
  scanf("%d/%d/%d", &d.dd, &d.mm, &d.yyyy);
  return d;
}
```

What's bad?

```
date_t dob= read_date();
```

Imagine a similar situation for

```
student_t student= read_student();
```

(recall that s student_t variable consumes 332 bytes)

Review: function for input, version 2 – The Good

```
void read_date(date_t *pd) {
  scanf("%d/%d/%d", &pd->dd, &pd->mm, &pd->yyyy);
}
```

How to use read_date? How good is this vesrsion: for date_t, for student_t ?

Review: function for input, version 2 – 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?

```
date_t dob;
read_date(&dob);

student_t stud;
read_stud(&stud)
```

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, for both input and output of a function.

Case Study: Polygons (Ex 8.2-8.4 combined & more)

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: Q&A

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
  - implementing Stage 1 [*incremental development*]
  - read, understand Stage 2 and implement it first
  - then, move on with stage 3. Check if you want to re-use anything from stage 2, then you should make that part into a function.
- 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 test today (and see if that compiler complains)