CSE216 Programming Abstractions

C Introduction

YoungMin Kwon

Hello World

- Create hello.c and implement the following

```
//TODO: include stdio.h

//TODO: write main function signature
//      - it takes no parameter
//      - it returns int

{
    //TODO: call printf with "Hello world!\n" message

    //TODO: return 0
}
```

Hello World

- Create hello.c and implement the following

```c
//TODO: include stdio.h
#include <stdio.h>

//TODO: write main function signature
//      - it takes no parameter
//      - it returns int
int main()
{
    //TODO: call printf with "Hello world!\n" message
    printf("Hello world!\n");

    //TODO: return 0
    return 0;
}
```

printf is declared in stdio.h

Program starts from main function

Returning 0 means a normal termination

To Compile and Execute

```
> gcc hello.c                                   [ or ]

> gcc -g -o hello hello.c
```

specify output

add debug info

```
> dir                                           [in Linux or Mac, ls]
…

05/02/2021  05:45 PM    <DIR>          .
05/02/2021  05:45 PM    <DIR>          ..
05/02/2021  05:36 PM            48,463 a.exe
05/02/2021  05:41 PM               231 hello.c
05/02/2021  05:45 PM            48,463 hello.exe
…

> a.exe                                         [in Linux or Mac, ./a.out]
factorial 4 = 24

> hello.exe                                     [in Linux or Mac, ./hello]
factorial 4 = 24
```

Greetings

- Create greeting.c and implement the following

```
//TODO: include stdio.h

//TODO: write main function signature
//      - it takes no parameter
//      - it returns int

{
    //TODO: define a variable name of an array of 100 chars


    //TODO: call printf with "Enter your name:" message


    //TODO: call scanf to read name. use "%99s" format


    //TODO: call printf with "Hello %s" and name
}
```

Greetings

- Create greeting.c and implement the following

```c
//TODO: include stdio.h
#include <stdio.h>

//TODO: write main function signature
//      - it takes no parameter
//      - it returns int
int main() {
    //TODO: define a variable name of an array of 100 chars
    char name[100];

    //TODO: call printf with "Enter your name:" message
    printf("Enter your name:");

    //TODO: call scanf to read name. use "%99s" format
    scanf("%99s", name);

   //TODO: call printf with "Hello %s" and name
   printf("Hello %s\n", name);
}
```

GCD

- Create gcd.c and implement the following

```c
#include <stdio.h>
//TODO: wirte the signature of gcd function
//    - it takes two integer parameters a and b
//    - it returns an integer

{
    //TODO implement gcd
}
int main() {
    int a, b;
    printf("GCD of a and b\n");
    printf("Enter a:");
    scanf("%d", &a);

    printf("Enter b:");
    scanf("%d", &b);

    printf("GCD of %d and %d = %d\n", a, b, gcd(a, b));
    return 0;
}
```

GCD

- Create gcd.c and implement the following

```
//TODO: wirte the signature of gcd function
//    - it takes two integer parameters a and b
//    - it returns an integer
int gcd(int a, int b)
{
    //TODO implement gcd
    if(a == b)
        return a;
    else if(a > b)
        return gcd(a - b, b);
    else
        return gcd(b - a, a);
}
```

Reverse String

- Create reverse.c and implement the following

```c
#include <stdio.h>
#include <string.h>
//TODO: wirte the signature of reverse function
//    - it takes a char pointer str
//    - it returns void (nothing)

{
    int i = 0;
    int j = strlen(str) - 1;
    //TODO: implement reverse
}
int main() {
    char str[100];
    printf("Enter a string: ");
    scanf("%99s", str);
    reverse(str);
    printf("result: %s\n", str);
    return 0;
}
```

Reverse String

- Create reverse.c and implement the following

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

//TODO: wirte the signature of reverse function
//    - it takes a char pointer str
//    - it returns void (nothing)
void reverse(char *str)
{
    int i = 0;
    int j = strlen(str) - 1;
    //TODO: implement reverse
    while(i < j) {
        char t = str[i];
        str[i] = str[j];
        str[j] = t;
        i++;
        j--;
    }
}
```

strlen is declared in string.h

Sort String

- Create sort.c and implement the following

```
//TODO: write the signature of find_min
//    - it takes a char* str and an integer from
//    - it returns an integer

{
    //TODO: implement find_min
    //    - it returns the index of the minimum element
    //      of str starting from index from
}

//TODO: write the signature of sort
//    - it takes a char* str
//    - it returns void (nothing)

{
    //TODO: implement sort
    //    - it sorts str using selecction sort
    //    - use find_min
}
```

Sort String

- Create sort.c and implement the following

```c
int main() {
    char str[100];
    printf("Enter a string: ");
    scanf("%99s", str);
    sort(str);
    printf("result: %s\n", str);
    return 0;
}
```

Sort String

- Create sort.c and implement the following

```
//TODO: write the signature of find_min
//    - it takes a char* str and an integer from
//    - it returns an integer
int find_min(char *str, int from)
{
    //TODO: implement find_min
    //    - it returns the index of the minimum element
    //      of str starting from index from
    int i, m = from;
    for(i = from+1; str[i]; i++)
        if(str[i] < str[m])
            m = i;
    return m;
}
```

C string ends with 0 (false)

Sort String

- Create sort.c and implement the following

```
//TODO: write the signature of sort
//    - it takes a char* str
//    - it returns void (nothing)
void sort(char *str)
{
    //TODO: implement sort
    //    - it sorts str using selecction sort
    //    - use find_min
    for(int i = 0; str[i]; i++) {
        int j = find_min(str, i);
        char t = str[i];
        str[i] = str[j];
        str[j] = t;
    }
}
```

Bisection

- Create bisection.c and implement the following

```c
/*
    bisection.c
*/
#include <stdio.h>
#include <stdlib.h>
#define EPSILON (1e-8)
/* fabs(x) returns the absolute value of x
*/
double fabs(double x) {
}
/* fx(x) returns x*x -2
*/
double fx(double x) {
}
```

Bisection

```c
/*
    bisection.c
*/
#include <stdio.h>
#include <stdlib.h>
#define EPSILON (1e-8)
/* fabs(x) returns the absolute value of x
*/
double fabs(double x) {
    return x < 0 ? -x : x;
}

/* fx(x) returns x*x -2
*/
double fx(double x) {
    return x*x - 2;
}
```

exit is declared in stdlib.h

```
/* bisection(f, a, b) returns x such that a <= x <= b and f(x) = 0
*/
double bisection(double (*f)(double), //function pointer
                 double a,
                 double b) {
    //m of double type is the middle point of a and b

    //if |a - b| < EPSILON, return m

    //else if f(a) * f(m) <= 0, search in the interval of [a, m]

    //else if f(m) * f(b) <= 0, search in the interval of [m, b]

    //otherwise, print "error" and exit

}
```

```c
/* bisection(f, a, b) returns x such that a <= x <= b and f(x) = 0
*/
double bisection(double (*f)(double), //function pointer
                 double a,
                 double b) {
    //m is the middle point of a and b
    double m = (a + b)/2;
    //if |a - b| < EPSILON, return m
    if (fabs(a - b) < EPSILON)
        return m;
    //if f(a) * f(m) <= 0, search in the interval of [a, m]
    else if (f(a) * f(m) <= 0)
        bisection(f, a, m);
    //if f(m) * f(b) <= 0, search in the interval of [m, b]
    else if (f(b) * f(m) <= 0)
        bisection(f, m, b);
    //otherwise, print "error" and exit
    else {
        printf("Error\n");
        exit(0);    //need stdlib
    }
}
```

```c
/* call bisection with fx, 0 and 2 to get ans
*/
int main() {
    double ans = bisection(fx/*function pointer*/, 0, 2);
    printf("ans: %lf\n", ans);
}
```

** compile and run **

```
$ gcc bisection.c
$ ./a.out
ans: 1.414214
```