CS 240: Programming in C

Lecture 10: Midterm Review

PURDUE UNIVERSITY

Announcements

- Midterm 1 is tomorrow!
  - CL50 224, MATH 175
  - 8:00 pm - 10:00 pm
  - DRC exam in MATH 215, 6:00 pm - ≤ 10:00 pm
  - If you have a conflict with another exam, email me
- Find your seat on the seating chart / maps
  - BEFORE coming to the exam
  - Let me know if you need an opposite-handed desk in MATH 175

Homework 3

```
291 scores total...
100+:  (0)
 100: ========================= (70)
  90: ======================== (68)
  80: ============== (41)
  70: ========== (33)
  60: === (8)
  50: === (9)
  40: == (6)
  30: == (6)
  20: = (1)
  10: = (2)
   0: ================ (47)
```

Review

- Questions on the exam may cover anything covered during lecture
- You are encouraged to:
  - Review lecture notes and videos
  - Hand write code
    - Quizzes
    - Lecture examples
    - Parts of homeworks
  - Practice writing quickly but clearly

Topics

- Compiling and linking
  - gcc options and usage
  - Object files and executables
- File operations
  - fopen() / fclose()
  - fprintf() / fscanf()
  - fseek() / ftell()
  - fread() / fwrite()
  - Error checking and error handling

Topics

- Typedef
  - Syntax, usage
- Structures
  - Properties
  - Declaration
  - Definition
  - Initialization
  - Nested structure declarations
  - Arrays of
  - Passing to and returning from functions
  - Assignment

Topics

- assert()
  - When should you use it?
- Basic string operations
  - strncpy()
  - strcmp()
  - What do they rely on for correctness?
- Variables
  - Are they global or local? Why?
  - Memory layout
    - Alignment and padding

Topics

- Variables
  - sizeof()
  - Arrays and their initialization
  - Endianness
- Unions
- Enums
- Bitwise operators
  - Masking
  - Bit flags

Compilation and Linking

- Write the command to compile a single C file named “hello.c” into an object file called “hello.o”.

```
$ gcc -c hello.c
```

Compilation and Linking

- Write the command to link two object files named “hello.o” and “goodbye.o” into the executable called “application”.

```
$ gcc -o application hello.o goodbye.o
```

Compilation and Linking

- Can you “run” an object file if it contains the “main()” function?

  No! It needs to be linked first

Compilation and Linking

- Can you “run” an executable that contains a single function called “main()”?

  Yes!

Compilation and Linking

- Can you “run” an executable that does not contain a function called “main()”?

Technically yes! But it’s uncommon

```
#include <stdio.h>
#include <stdlib.h>

int print_hello() {
     printf("Hello!\n");
     return 0;
}

void _start() {
     exit(print_hello());
}
```

```
$ gcc -nostartfiles hello.c
$ ./a.out
Hello!
```

Compilation and Linking

- What does the “-Wall” flag do?

  Enables ~~all~~ most warnings

Compilation and Linking

- What does the “-g” flag do?

  Compiles with debug symbols

Compilation and Linking

- What does the “-ansi” flag do?

  Compiles according to the ANSI standard (i.e., C90)
  - Disables C++-style // comments
  - Other various changes

Compilation and Linking

- What does the “-c” flag do?

  Compiles a source file into an object file

  Does not link

Compilation and Linking

- What does the “-o” flag do?

  Names the output file

  Without “-o”, the file is named:

  - “a.out” if linking into an executable
  - “X.o” if using the “-c” flag
    - where X is the name of the source file
    - e.g. hello.c becomes hello.o

File Operations

- Given the following FILE pointer variable definition, write the code that will open a file named “hello.txt” for read-only access and print a message of your choice if there was an error in doing so.

```
FILE *my_file = 0;
my_file = fopen(“hello.txt”, “r”);
if (!my_file) {
  printf(“Goodbye!\n”);
}
```

File Operations

- Write code that will, without opening any file, check if a file named “hello.txt” can be opened for read access. Put the code inside the ‘if’ predicate:

```
if (access(“hello.txt”,R_OK) == 0) {
  /* Yes, we can open the file... */
}
```

File Operations

- Write code that will, without opening any file, check if a file named “hello.txt” can be opened for write access. Put the code inside the ‘if’ predicate:

```
if (access(“hello.txt”,W_OK) == 0) {
  /* Yes, we can open the file... */
}
```

Typedef

- Declare a type called “my_array_t” that is an array of 15 floats.

```
typedef float my_array_t[15];
```

Typedef

- Declare a type called “struct_arr_t” that is an array of 10 structs of the format:

```
typedef struct str {
  int x;
  int y;
} struct_arr_t[10];
```

Typedef

- Define a variable called my_str_arr of type struct_arr_t

```
struct_arr_t my_str_arr;
```

Structures

- Can two elements within a structure have the same name?

  No!

Structures

- Can you initialize a structure like this?

```
struct my_str {
  int x;
  float y;
} mine = { 0, 0.0 };
```

Yes!

Structures

- Can you initialize a structure like this?

```
struct my_str {
  int x;
  float y;
};
void my_func(int n) {
  my_str mine = { n, 0.0 };
}
```

No!

Structures

- Declare a structure that contains an integer element named i, a floating point element named f, and an array of 20 characters named str (in that order). Name it anything you want.

```
struct my_struct {
  int i;
  float f;
  char str[20];
};
```

Structures

- Define a variable called “my_new_struct” of the type in the previous question.

```
struct my_struct my_new_struct;
```

Structures

- Define a variable called “my_array_of_structs” that is an array of 40 structures of the type in the prior two questions.

```
struct my_struct my_array_of_structs[40];
```

Assert

- Under what circumstances would you place an assert() into your code?

  - When an unrecoverable error occurs
  - To double-check your assumptions / expectations

Assert

- What will be the result of the following code?

```
int my_func() {
  int count = 0;
  int sum = 0;
  for (count = 0; count < 100; count++) {
    assert(sum > 0);
    sum = sum + count;
  }
  return sum;
}
```

Assertion will fail on the first loop (sum = 0)

Assert

- What might you do to the previous code to make it do a “better” job?
    - Change it to assert(sum >= 0);

Variables

- What is the difference between initialization of a variable and assignment to a variable?

  - Initialization creates a variable and gives it a value at the same time
  - Assignment replaces an existing variable with a new value

Variables

- What is the difference between a declaration and a definition?
    - A declaration announces the properties of a type or function
        - “describes it”
    - A definition allocates storage for a variable or function

Variables

- What is the difference between a global variable and a local variable?

  - A global variable is created outside of a function
    - Accessible to the entire program
    - Never goes out of scope
    - Initialized to zero
  - A local variable is created inside of a function
    - Can only be accessed inside that function
    - Not automatically initialized

Variables

- What is the size of:

```
struct my_coord {
  int x;
  int y;
  double altitude;
};

sizeof(struct my_coord) = ?
```

16

Variables

- What is the size of:

```
struct my_line {
  struct my_coord first;
  struct my_coord second;
  char name[10];
};

sizeof(struct my_line) = ?
```

48

Variables

- What is the size of:

```
struct my_coord var;
struct my_coord array[3];
struct my_line one_line;
struct my_line two_lines[2];
```

```
sizeof(var)       = ?  16
sizeof(array[1])  = ?  16
sizeof(array[2])  = ?  16
sizeof(array)     = ?  48
sizeof(two_lines) = ?  96
sizeof(one_line)  = ?  48
```

Variables

- How many bytes large is the following definition?

```
struct my_coord new_array[] = {
  { 0,0,3.5 },
  { 1,2,4.5 },
  { 2,0,9.5 }
};
```

48