C: structs; Number representation: binary

COSC 208, Introduction to Computer Systems, 2021-09-08

Announcements

- Project 1 out today; read for next lecture
- Part A normally due next Thursday at 11pm

Outline

- Warm-up
- Structs
- Binary

Warm-up

Q1: *Write a function called `count_words` that takes a string and counts the number of words in the string. Assume each word is separated by a single space, and the string will contain at least one word. For example, `"Today is Wednesday."` contains 3 words.*

```
int count_words(char str[]) {
    int words = 1;
    for (int i = 0; i < strlen(str); i++) {
        if (nums[i] == ' ') {
            words++;
        }
    }
    return words;
}
```

Structs

- *How is a struct declared?*

  ```
  struct tvshow {
      char name[100];
      int season;
  };
  ```

- *How are fields of the struct accessed?* — with the dot (.) operator

  ```
  struct tvshow favorite;
  strncpy(favorite.name, "Tiny House Nation", 100);
  ```

```
favorite.season = 6;
```

- A struct is a collection of values; it is **not** an object, and hence cannot have methods associated with it
- A struct variable holds *values* for the fields of the struct; it is **not** a reference to the struct, and hence a copy of the values are passed to functions
- Q2: *What is the output of this program?*

```c
struct one {
    char x;
    char y;
    short z;
};
struct two {
    int m;
    int n[10];
};
int main() {
    struct one a;
    struct two b;
    printf("%d %d\n", sizeof(struct one), sizeof(a.z));
    printf("%d %d\n", sizeof(b), sizeof(b.n));
}
```

```
4 2
44 40
```

- Q3: *What is the output of this program?*

```c
struct alpha {
    char x[10];
    int y;
};
struct beta {
    int b;
    int c;
};
int main() {
    struct alpha a = { "Colgate", 13 };
    struct beta b = { 1, 2 };
    struct beta c = { 3, 4 };
    a.y += -13;
    b.b = 5;
    c = b;
    b.c = 6;
    printf("a %s %d\n", a.x, a.y);
```

```
    printf("b %d %d\n", b.b, b.c);
    printf("c %d %d\n", c.b, c.c);
}
```

```
a Colgate 0
b 5 6
c 5 2
```

- Q4: *Draw the stack right before the return from* `mystery`

```
struct personT {
  char name[32];
  int  age;
};

void mystery(int i_val, struct personT per, int a[], int n);

int main() {
  struct personT person;
  int x, i;
  int arr[5];

  for(i=0; i < 5; i++) {
     arr[i] = i;
  }
  x = 13;
  strcpy(person.name, "Lila");
  person.age = 10;
  mystery(x, person, arr, 5);

  for(i=0; i < 5; i++)
     printf("arr[%d] = %d\n", i, arr[i], 5); // bucket val: 0 1 4 9 16

  printf("x = %d age = %d name = %s\n", x, person.age, person.name);
}               // values:  13 10 Lila

void mystery(int i_val, struct personT per, int a[], int n) {
  for(int i=0; i < n; i++) {
     a[i] = a[i]*a[i];
  }
  strcpy(per.name, "Orso");
  per.age = 18;
  i_val = 100;
  //**** DRAW STACK IS RIGHT BEFORE return STATEMENT IS EXECUTED
  return;
}
```

Passing a struct, a statically declared array, and two ints to a function. All arguments are passed by value.

- The value of the array name argument is the base address of the array argument thus
    - parameter refers to the same array buckets as the argument
    - function can modify the bucket values in the passed array
- However, the function cannot modify any values of the struct or int arguments.

Decimal (i.e., base 10)

- *How many unique values can you represent with one decimal digit?* -- 10
- *How do you count to 13 (in base 10)?* -- 0 (computers like to start at zero!), 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 (notice the "ones" place becomes zero and the "tens" place becomes 1), 11, 12, 13

Binary (i.e., base 2)

- We are used to working with decimal numbers, but computers represent numbers in binary
    - A single binary digit (*bit*) can represent two unique values — 0 or 1
    - It is easier to build hardware that uses polar opposites (e.g., on/off, high/low voltage, positive/negative magnetism, etc.) than multiple levels (e.g., very high, high, moderately high, moderate, moderately low, low, very low voltage)
- *How many unique values can be represented with 2 bits?* — 00, 01, 10, 00; 2^2 = 4
    - *3 bits?* — 000, 001, 010, 011, 100, 101, 110, 111; 2^3 = 8
- Practice powers of two: 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192
- *How many bits are in a byte?* -- 8 bits
- _How many unique values can be represented with 1 byte? 4 bytes? -- 256, ~4.3 billion
- *How do you count to 13 in binary?* -- 0, 1, 10, 11, 100, 101, 110, 111, 1000, 1001, 1010, 1011, 1100, 1101
- *Convert these binary numbers to decimal (i.e., base 10):*
    - Q5: `0b10` = 1 * 2^1 + 0 * 2^0 = 2
    - Q6: `0b11` = 1 * 2^1 + 1 * 2^0 = 2 + 1 = 3
    - Q7: `0b1010` = 1 * 2^3 + 0 * 2^2 + 1 * 2^1 + 0 * 2^0 = 8 + 2 = 10
    - Q8: `0b1111` = 1 * 2^3 + 1 * 2^2 + 1 * 2^1 + 1 * 2^0 = 8 + 4 + 2 + 1 = 15
    - Q9: `0b11001100` = 1 * 2^7 + 1 * 2^6 + 0 * 2^5 + 0 * 2^4 + 1 * 2^3 + 1 * 2^2 + 0 * 2^1 + 0 * 2^0 = 128 + 64 + 8 + 4 = 204

Extra practice

- Q10: *Write a struct definition to represent a date (year, month number, and day).*

```
struct date {
    int year;
    int month;
    int day;
};
```

- Q11: *Write a function called `compare` that takes two date structs and returns -1 if the first date occurs before the second, 0 if the dates are equal, and 1 if the first date occurs after the second.*

```
int compare(struct date a, struct date b) {
    if (a.year < b.year) {
        return -1;
    } else if (b.year < a.year) {
        return 1;
    } else {
        if (a.month < b.month) {
            return -1;
        } else if (b.month < a.month) {
            return 1;
        } else {
            if (a.day < b.day) {
                return -1;
            } else if (b.day < a.day) {
                return 1;
            } else {
                return 0;
            }
        }
    }
}
```