Program memory: arrays & pointers; malloc

COSC 208, Introduction to Computer Systems, 2022-02-22

Announcements

- Exam 1 this Thursday

Warm-up

- Q1: *What is the output of this program?*

```
void increment1(int a) {
    a = a + 1;
}

void increment2(int *b) {
    *b = *b + 1;
}

int main() {
    int x = 1;
    int *y = &x;
    increment1(x);
    printf("%d %d\n", x, *y);
    increment2(y);
    printf("%d %d\n", x, *y);
}
```

Stack memory layout

- Q2: *What is the output of this program?*

```
int main() {
    int a = 1; // Assume at 0x4
    int *x = &a; // Assume at 0x8
    int **y = &x; // Assume at 0xC
    printf("%p %p %p\n", a, x ,y);
    printf("%p %p\n", *x , *y);
}
```

```

```

Arrays & pointers

Example 1

```
int main() {
    char word[] = "hat";
    printf("word = %s\n", word);
    char *ptr = word;
    printf("ptr = %s\n", ptr);
    if (ptr == word) {
        printf("ptr == word\n");
    }
    else {
        printf("ptr != word\n");
    }
    word[1] = 'i';
    printf("word = %s\n", word);
    *ptr = 's';
    printf("word = %s\n", word);
    ptr[1] = 'a';
    printf("word = %s\n", word);
}
```

Example 2

```
int update(char str[]) {
    str[0] = 'p';
}
int main() {
    char word[] = "mat";
    update(word);
    printf("%s\n", word);
}
```

Practice

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

```
int main() {
    int nums[4] = {1,2,3,4};
    printf("%d %d\n", *nums, nums[1]);
    int *ptr = &nums[1];
    nums[1] += 4;
    printf("%d %d\n", *ptr, nums[0]);
    ptr = (nums + 2);
    printf("%d\n", *ptr);
    ptr++; // num++ is illegal
    printf("%d\n", *ptr);
}
```

- Q4: *What is the output of this program?*

```
int main() {
    char *first = "Colgate";
    char second[10] = "Univ";
    char *f = &first[3];
    printf("%d\n", strlen(f));
    char *s = second;
    *s = 'K';
    s++;
    *(s+2) = 't';
    printf("%s %s\n", second, s);
}
```

Pointers as return values

- *What happens?*

  ```
  int *one() {
      int x = 1;
      int *p = &x;
      return p;
  }
  int main() {
      int *q = one();
      printf("%d\n", *q);
  }
  ```

Program memory

- Q5: *Write a function called `duplicate` that takes a string (i.e., an array of `char`) as a parameter and returns a copy of that string stored on the heap.*

```
```

- Q6: *Write a function called `range` that behaves similar to the `range` function in Python. Your function should take an unsigned integer (`length`) as a parameter, and return a dynamically allocated array with `length` unsigned integers. The array should be populated with the values 0 through `length-1`.*

Q7: *Draw a memory diagram that displays the program's variables and their values when the program reaches the comment* `STOP HERE`.

```
char *split(char *str, char delim) {
    for (int i = 0; i < strlen(str); i++) {
        if (str[i] == delim) {
            str[i] = '\0';
            return &str[i+1];
        }
    }
    return NULL;
}

void parse(char *url) {
    char separator = '/';
    char *path = split(url, separator);
    int domainlen = strlen(url);
    int pathlen = strlen(path);
    // STOP HERE
    printf("Domain (%d chars): %s\n", domainlen, url);
    printf("Path (%d chars): %s\n", pathlen, path);
}

int main() {
    char input[] = "colgate.edu/lgbtq"
    parse(input);
}
```

Q8: *What do the following two functions do? How are they different?*

```
void swap1(int *m, int *n) {
    int tmp = *n;
    *n = *m;
    *m = tmp;
}
void swap2(int **x, int **y) {
    int *tmp = *y;
    *y = *x;
    *x = tmp;
}
```

Extra practice

- QA: *Write a function called `generate_password` that takes an unsigned integer (`length`) as a parameter, and returns a dynamically allocated array of with `length` randomly selected characters (e.g., uppercase letters, lowercase letters, digits, symbols). Your function should use the `rand()` function from the C standard library, which returns a pseudo-random integer in the range 0 to `RAND_MAX`.*

- QB: *Write a function called `substring` that takes a string, a starting index, and a length, and returns a substring. If the starting index is too large, the function should return `NULL`. If the length is too large, the function should return a shorter substring.*

- QC: *Write a function called `lengths` that takes an array of strings and the number of elements in the array and returns an array of integers containing the length of each string.*