APS105 Lecture 12
5.8, Review

Dan Zingaro

February 22, 2010

Organizing Larger Programs

- We do not have to stuff all of our functions into the same `.c` file
- What we can do is create multiple `.c` files, each with related function definitions
- For each `.c` file, we create a `.h` file that contains the prototypes for some of the functions in that `.c` file
- We only write prototypes in the `.h` file for the functions we want other `.c` files to be able to use
- Why do this?
  - Makes the organization of the program clear
  - Allows us to more easily reuse functions in other programs

Example: Program Organization

- We often use random-number functions in our code
  - A function to initialize the random-number generator
  - A function to get a random integer between lower bound `i` and upper bound `j`
- If we create a `.c` and a `.h` file for these functions, we can easily use them in our programs
- `randfuncs.h` gives the prototypes for the functions that we will write
- `randfuncs.c` gives the function definitions

Example: Program Organization... (randfuncs.h)

```
/*Initialize random number generator*/
void initRand (void);

  /*Return a random integer in the range i..j*/
int randBetween (int i, int j);
```

Example: Program Organization... (randfuncs.c)

```
#include <stdlib.h>
#include <time.h>
#include "randfuncs.h"

/*Initialize random number generator*/
void initRand (void) {
  srand (time(0));
}

/*Return a random integer in the range i..j*/
int randBetween (int i, int j) {
  int nums = j - i + 1;
  return rand() % nums + i;
}
```

Example: Program Organization... (randmain.c)

We can use these functions to write a program that rolls three dice.

```
#include <stdio.h>
#include "randfuncs.h"

int main (void) {
  initRand();
  for (int i = 1; i <= 3; i++)
    printf ("Rolled a %d\n", randBetween (1, 6));
  return 0;
}
```

Review Questions

ConcepTest

```
int player;
```

Assume `player` has value 1 or 2; if it has value 1, we want to change it to 2, and if it has value 2, we want to change it to 1. Which of the following code fragments will do this?

- A. `player = !player;`
- B. `player = player % 2;`
- C.

```
if (player == 1)
  player = 2;
if (player == 2)
  player = 1;
```

- D. More than one of the above
- E. None of the above

ConcepTest

```
char ch;
```

Which of the following code fragments could be used to ensure that we get a `y` or `n` response?

- A.

```
do {
  ...
} while (ch != 'y' && ch != 'n');
```

- B.

```
do {
  ...
} while (ch != 'y' || ch != 'n');
```

- C.

```
do {
  ...
} while (!(ch == 'y' || ch == 'n'));
```

- D. More than one of the above
- E. None of the above

ConcepTest

In which order will the following fragments result in a program that asks for five **positive** integers and prints their maximum?

```
(1) scanf ("%d", &num);
(2) if (num > max)
(3) printf ("%d\n", max);
(4) max = num;
(5) int num;
(6) }
(7) int max = 0;
(8) for (int i = 1; i <= 5; i++) {
```

- A. (5), (7), (1), (2), (4), (8), (1), (2), (4), (6), (3)
- B. (5), (7), (8), (1), (2), (4), (6), (3)
- C. (5), (7), (1), (2), (8), (1), (2), (4), (6), (3)
- D. (8), (5), (7), (1), (2), (4), (6), (3)
- E. None of the above