C Exercises

Ben Vandiver

February 28, 2003

1 General Information

These exercises are intended for you to practice on. Since they won't be graded, they are to help you practice the types of things you will need in the remainder of the class. The scratch exercises are designed to simply test your knowledge of the syntax and the basics of the semantics of various C constructs. Because they are intended for quick-response, the solutions will be provided at the end of this document. The interesting exercises that follow will have solutions that can be requested when you think you have a solution or when you get horribly stuck. The reason they won't be just made available is to encourage you to work a bit before seeing the solution.

2 Scratch Exercises

For these exercises, you should start from scratch (blank C file) each time, preferably without someone else helping you (unless you get stuck).

Write a program that:

1. prints `Hello World`.
   See `printf`
2. reads a number from the user and then prints it out
   See `scanf`
3. reads a number, then prints the integers from 1 to the number.
   See *for loops*
4. reads a number, then prints the integers from 1 to the number to a file.
   See *file operations*,`fopen`,`fprintf`,`fclose`
5. creates an array, where the i^{th} element is equal to i^2. Print the values from the array in a separate loop.
   See *array operations*
6. write `print3`, a function which prints its 3 integer arguments.
   See *functions*

7. write `twotimesn` which returns an integer which is twice its input. Have your main program call the function with a couple of numbers and print the function's responses. The `twotimesn` function should contain no calls to `printf`.
   See *functions*,`return`

3 Interesting Exercises

These exercises are intended to sharpen your programming skills in two ways: give a firmer handle on syntax through more practice, and encourage development of the problem-statement-to-C-process mental translation that must occur before a program can be written. Each problem consists of a problem statement, a set of hints, and possible extensions should you want to "go nuts." Sometimes the problem statement will be intentionally vague, leaving you room to decide how to characterize your solution. Solutions are available upon request.

3.1 Guess A Number

3.1.1 Problem Description

Write a program that picks a random number between 1 and 100 inclusive. It then prompts the user to guess the number. If the user guesses the number, print out a congratulatory message and exit. Otherwise, indicate whether the guess was low or high and give the user an opportunity to guess again, repeating the process with their new guess.

3.1.2 Hints

Think `while` loop. In order to select a random number, you must do a little setup. Use the following as the kernel of your program:

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

int main(void) {
  // variable declarations here
  int num;

  srand(time(NULL));  // initialize random number generator

  num = (rand()%100)+1; // get random number

  // guessing bits here

  return 0;
}
```

3.1.3 Extensions

Flip the problem around and ask the user to select a number. Then have the program make a guess and ask the user if it's correct, high or low. (You can have the user respond with 0 for correct, 1 for low, 2 for high, or some other scheme of assigning numbers to responses; string comparison is a little odd). Come up with a couple types of "guessing strategies" for the program. What would be the simplest (and slowest) solutions? Can you write the optimal strategy?

4 Solutions to scratch exercises

1. print `Hello, World`

```
#include <stdio.h>
int main(void) {
  printf("Hello, World");
  return 0;
}
```

2. read a number, print it out

```
#include <stdio.h>
int main(void) {
  int x;
  printf("Enter a number: ");
  scanf("%d",&x);
  printf("The number is %d\n",x);
  return 0;
}
```

3. read a number, print 1 to n

```
#include <stdio.h>
int main(void) {
  int n,i;
  printf("Enter a number: ");
  scanf("%d",&n);
  for (i=1; i <= n; i++) {
    printf("%d\n",i);
  }
  return 0;
}
```

4. read a number, print 1 to n to a file

```
#include <stdio.h>
int main(void) {
  int n,i;
  FILE *f;

  f = fopen("output.txt","w");
  if (f == NULL) {
    printf("Unable to open file for output!\n");
    return -1;
  }
```

```
  printf("Enter a number: ");
  scanf("%d",&n);
  for (i=1; i <= n; i++) {
    fprintf(f,"%d\n",i);
  }
  fclose(f);
  return 0;
}
```

5. create an array of squares, print array

```
#include <stdio.h>
int main(void) {
  int arr[10];
  int i;

  // initialize array
  for (i=0; i < 10; i++) {
    arr[i] = i*i;
  }

  // print array
  for (i=0; i < 10; i++) {
    printf("%d\t",arr[i]);
  }
  printf("\n");
  return 0;
}
```

6. write `print3` which prints its arguments

```
#include <stdio.h>

void print3(int x, int y, int z);

int main(void) {
  print3(1,2,3);
  print3(-1,0,1);
  return 0;
}

void print3(int x, int y, int z) {
  printf("Arg 1: %d\nArg 2: %d\nArg 3: %d\n",x,y,z);
}
```

7. write `twotimesn`

```
#include <stdio.h>

int twotimesn(int n);

int main(void) {
  int x;
  x=21; printf("twotimesn(%d) = %d\n",x,twotimesn(x));
  x=-5; printf("twotimesn(%d) = %d\n",x,twotimesn(x));
  return 0;
}

int twotimesn(int n) {
  return 2 * n;
}
```