Input/Output Practice with printf and scanf

README

- This file is a practice file for the C output function `printf`.
- Time: approx. 30-60 min.
- When you're done with a section move the cursor on the section heading and type `S-<right>` (or SHIFT+ <right-arrow>).

TODO Identify yourself

- replace the placeholder `[yourName]` in the header of this file by your name and save the file (`C-x C-s`).

printf

DONE Conversion specification

Recreate the output below exactly, using only format specifiers (no extra white space).

```
: ....|....|....|
:   100100
: 200200
:        3.1416
: 3.141593
```

—— SOLUTION ——

```
printf("....|....|....|\n");
printf("%8d\n", 100100);
printf("%-10d\n", 200200);
printf("%13.4f\n", 3.141593);
printf("%-.6f\n", 3.141593);
```

```
....|....|....|
  100100
200200
       3.1416
3.141593
```

DONE Integer decimal `d`

Show that the default for `d` is p=1. Print the numbers 1, 1, 100 and 10000 with the specifiers `%d`, `%.1d`, `%.5d`, `%.2d`. Print each expression on its own line, but use only ONE `printf` statement.

—— SOLUTION ——

```
printf("....|....|....|\n");
```

```
printf("%d\n%.1d\n%.5d\n%.2d\n", 1, 1, 100, 10000);
```

```
....|....|....|
1
1
00100
10000
```

DONE Integer decimal precision p

Print the number 42 on a space of 10 characters with precision 5.

The result should look like this:

```
: ....|....|....|
:      00042
```

—— SOLUTION ——

#+name spec1

```
printf("....|....|....|\n");
printf("%10.5d\n", 42);
```

```
....|....|....|
     00042
```

DONE Scientific notation e

- Print 1, 1000.100, and 1,000,000,000,000,000 using `%e`.
- Provide for the required number of decimal positions (but not more)
- Print each expression on its own line with its own `printf` function.
- Add the header-argument `:results output` to the code block

Desired output:

```
: 1e+00
: 1.0001e+03
: 1e+15
```

—— SOLUTION ——

```
printf("%1.e\n", 1.);
printf("%.4e\n", 1000.1);
printf("%.e\n", 1000000000000000.);
```

```
1e+00
1.0001e+03
1e+15
```

DONE Variable floating point `g`

- Use the format specifier `g` to display the following numbers: `200`, `3.142574654` with p=8, `2.71`, and `!5`.
- print each on a line of its own, but use only **one** `printf` statement to do it!
- `!N` is defined as the factorial of `N`.

—— USE THIS CODE BLOCK ——

```
printf("....|....|....|\n");
...
```

—— SOLUTION ——

```
printf("....|....|\n");
printf("%g\n%.8g\n%g\n%g\n", 200., 3.142574654, 2.71, 5.*4.*3.*2.*1.);
```

```
....|....|
200
3.1425747
2.71
120
```

scanf

Scan integer and floating-point input

1. Use the code block 1 below for practice
2. Define two *integer* variables `k`, `l`, and two *floating-point* variables `u` and `v`
3. Complete the `scanf` *format string* and enter the variables list to scan these variables

4. Run the code block 1 below to generate an input file `scanf_input` (the input should **not** contain the `f` character).

   ```
   echo "100 -1000 .456 -9.34e2" > scanf_input
   cat scanf_input
   ```

5. Run the code block 1 to get the output:

   ```
   : |  100|-1000|0.456| -934|
   ```

```
// declare variables
...

// scan input
scanf("...", ...);

// print scanned input
printf("|...|...|...|...|\n", ...);
```

—— SOLUTION ——

```
// declare variables
int k, l;
float u, v;

// scan input
scanf("%d%d%f%f", &k, &l, &u, &v);

// print scanned input
printf("|%5d|%5d|%5.3f|%5.0f|\n", k, l, u, v);
```

Scanning ordinary characters

1. Run the C code block below with two input files, `ord1` and `ord2`.
2. Create the input files here:
   - the input file `ord1` contains `•5/•96` and should succeed
   - the input file `ord2` contains `•5 /•96` and should fail

   Create input file `ord1`:

   ```
   echo "..." > ord1
   ```

   Create input file `ord2`:

   ```
   echo "..." > ord2
   ```

3. Run program the program twice:
   - `ord1` as input file
   - `ord2` as input file

   Change the `#+name` of the program accordingly so that you can see both outputs next to each other (from `pgm:ordTest1` to `pgm:ordTest2`).

```
int i,j;

scanf("%d/%d", &i, &j);

printf("|%5d|%5d|\n", i, j);
```

—— SOLUTION ——

```
echo " 5/ 96" > ./data/ord1
```

```
echo " 5 / 96" > ./data/ord2
```

```
int i,j;

scanf("%d/%d", &i, &j);
```

```
printf("|%5d|%5d|\n", i, j);
```

```
int i,j;

scanf("%d/%d", &i, &j);

printf("|%5d|%5d|\n", i, j);
```

Match input patterns exactly

1. Run the code 1 below. It creates an input file `numbers` that contains: `444==++//555`

   ```
   echo "444==++//555" > numbers
   cat numbers
   ```

2. Complete the code 1 below to pick up only the numbers in the input file.

   ```
   int foo, bar;

   scanf(...)
   printf("%d %d", foo,  bar);
   ```

   —— SOLUTION ——

   ```
   int foo, bar;

   scanf("%d==++//%d", &foo, &bar);
   printf("%d %d", foo,  bar);
   ```

Add fractions

1. The program 1 prompts the user to add two fractions and then display their sum.

   Sample output for the input 5/6 and 3/4:

   ```
   5/6 + 3/4 = 38/24
   ```

2. Run the code block 1 to create the input file with the sample numbers.

   ```
   echo "5/6" > addFrac_input
   echo "3/4" >> addFrac_input
   cat addFrac_input
   ```

3. Complete the format strings below so that the program runs as intended!

   ```
   // declare variables
   int num1, denom1, num2, denom2, result_num, result_denom;

   // scan input
   ```

```
scanf("...", &num1, &denom1);
scanf("...", &num2, &denom2);

// compute numerator and denominator
result_num = num1 * denom2 + num2 *denom1;
result_denom = denom1 * denom2;

// print result
printf("%d/%d + %d/%d = %d/%d\n",
       num1, denom1, num2, denom2,
       result_num, result_denom);
```

1. Modify the program 1 so that there is only **on** `scanf` statement. Make sure that the modified program yields the same result as before.

—— SOLUTION ——

```
// declare variables
int num1, denom1, num2, denom2, result_num, result_denom;

// scan input
scanf("%d/%d", &num1, &denom1);
scanf("%d/%d", &num2, &denom2);

// compute numerator and denominator
result_num = num1 * denom2 + num2 *denom1;
result_denom = denom1 * denom2;

// print result
printf("%d/%d + %d/%d = %d/%d\n",
       num1, denom1, num2, denom2,
       result_num, result_denom);
```

```
5/6 + 3/4 = 38/24
```

```
// declare variables
int num1, denom1, num2, denom2, result_num, result_denom;

// scan input
scanf("%d/%d%d/%d",
      &num1, &denom1, &num2, &denom2);

// compute numerator and denominator
result_num = num1 * denom2 + num2 *denom1;
result_denom = denom1 * denom2;

// print result
printf("%d/%d + %d/%d = %d/%d\n",
       num1, denom1, num2, denom2,
       result_num, result_denom);
```

Author: [yourName] (pledged)
Created: 2024-02-21 Wed 10:44
Validate