Formatted I/O: scanf

CSC 100 Introduction to programming in C/C++

README

- There is much more to `scanf` and `printf` than we've seen
- I/O is where the pedal hits the metal - where man meets machine
- In this notebook: conversion specifications for scanf

Overview

- A `scanf` format string may contain ordinary characters and conversion specifications like `d`, `e`, `f`, `g`
- The conversions allowed with `scanf` are essentially the same as those used with `printf`
- The `scanf` format string tends to contain only conversion specs
- `[ ]`

  What will this sample input assign to the variables in 1 below?

```
1  -20  .3   -4.0e3
```

```
int i, j;
float x, y;

scanf("%d%d%f%f", &i, &j, &x, &y);

printf("|%5d|%5d|%5.1f|%5.1f|\n", i, j, x, y);
```

Main traps

- The compiler will not check that specs and input match
- The & symbol may not miss in front of the input variable

How scanf works

- `scanf` tries to math input groups with specs
- For each spec, it tries to locate an item in input
- It reads the item, and stops when it can't match
- If an item is not read successfully, `scanf` aborts

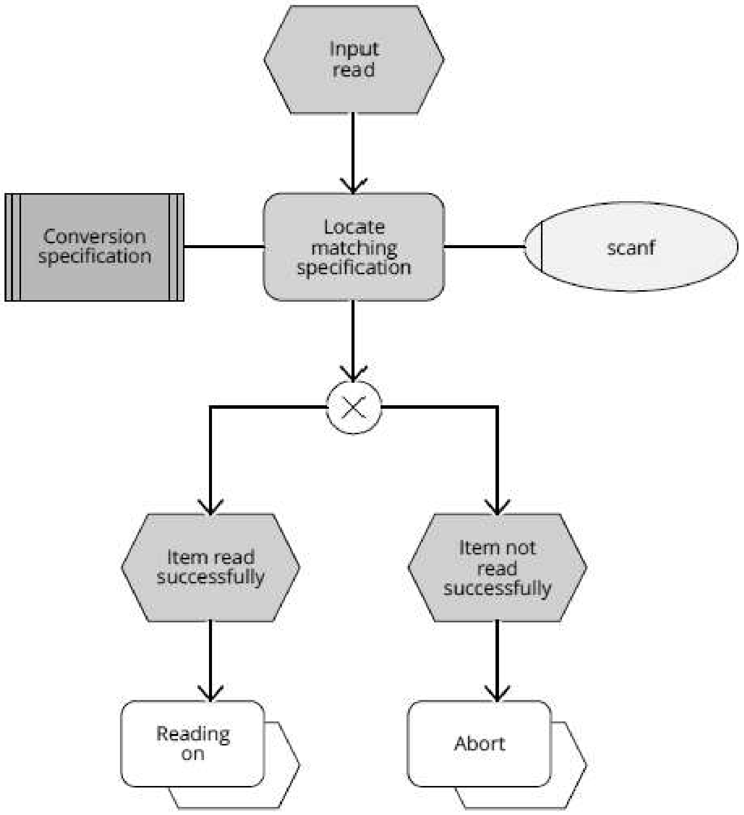
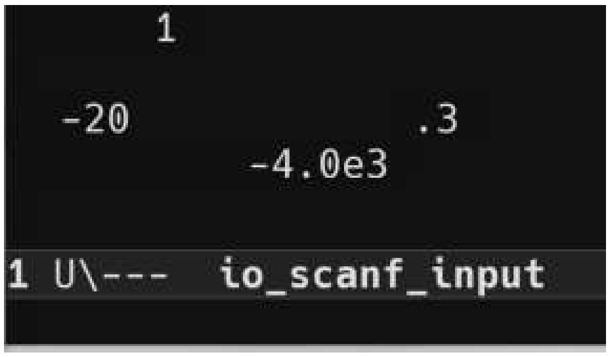


Figure 1: How scanf works

- White-space characters are ignored: SPC, TAB, new-line
- In 1 above, the lines can be on one line or spread over several lines:



```
    1

 -20            .3
       -4.0e3

1 U\---  io_scanf_input
```

Figure 2: Input file for tscanf

- `scanf` sees a character stream (¤ = new-line, s=skip'd, r= read):

```
••1¤-20•••.3¤•••-4.0e3¤
ssrsrrrsssrrssssrrrrrr
```

- When asked to read an integer (`%d` or `%i`), `scanf` searches for a digit, +/- sign, then reads until a non-digit
- When asked to read a float (`%f`, `%g`, `%e`), `scanf` looks for +/- sign, digits, decimal point, exponent (`e+02`, `e-02`)
- When used with `scanf`, `%e`, `%f`, `%g` are interchangeable
- When it finds a character that cannot be part of the current item, the character is returned to be read again during the scanning of the next input item or the next call of `scanf`

- The extended example below has the same spec as 1 - `"%d%d%f%f",&i,&j&x&y`

```
1-20.3-4.0e3¤
```

1. `%d`. Stores `1` in `i`, returns `-`
2. `%d`. Stores `-20` in `j`, returns `.`
3. `%f`. Stores `0.3` in `x`, returns `-`
4. `%f`. Stores `-4.0 x 10^3` in `y`, returns `¤`

Ordinary characters in format strings

- `scanf` reads white-space until it reaches a symbol
- When it reaches a symbol, it tries to match to next input
- It now either continues processing or aborts

- Example:

  If the format string is `"%d/%d"` and the input is `•5/•96`, `scanf` succeeds.

  If the input is `•5•/•96` , `scanf` fails, because the `/` in the format string doesn't match the space in the input.

- To allow spaces after the first number, use `"%d /%d"` instead
- `[ ]`

  Let's try it. Run the block 1 first with two input files:

  - the input file `ord1` contains `•5/•96` and should succeed
  - the input file `ord2` contains `•5 /•96` and should fail

```
int i,j;

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

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

```
    5   96
```

- `[ ]`

  Next, fix the `scanf` format string below to allow input from `ord2`:

```
int i,j;

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

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

```
    5   96
```

Confusing printf with scanf

- Calls to these only appear similar but they aren't
- Common mistakes:

1. putting `&` in front of variables in a `printf` call

```
printf("%d %d\n", &i, &j);  /*** WRONG ***/
```

2. assuming that `scanf` should resemble `printf` formats

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

   - After storing `i`, `scanf` will try to match a comma with the next input character. If it's a SPC, it will abort.x
   - Only this input will work: `100, 100` but not `100 100`

3. putting a `\n` character at the end of `scanf` string

```
scanf("%d\n", &i);
```

   - To `scanf`, the new-line is a SPC. It will advance to the next white-space character
   - This can cause the program to hang (wait forever for input)

Get coding: sample program

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

  Sample output:

```
Enter first fraction: 5/6
Enter second fraction: 3/4
The sum is 38/24
```

- `[ ]`

  Complete the format strings below so that the program runs as intended! The sample input is already stored in the `addfrac_input` file in the format shown.

```
int num1, denom1, num2, denom2, result_num, result_denom;

printf("Enter first fraction: ");
scanf("%d/%d", &num1, &denom1);

printf("Enter second fraction: ");
scanf("%d/%d", &num2, &denom2);

result_num = num1 * denom2 + num2 *denom1;
result_denom = denom1 * denom2;

printf("\nThe sum is %d/%d\n",result_num, result_denom);
```

```
Enter first fraction: Enter second fraction:
The sum is 38/24
```

```
Enter first fraction: Enter second fraction:
The sum is 38/24
```

Author: Marcus Birkenkrahe

Created: 2022-03-09 Wed 08:10

Validate