printf() and scanf()

CMSC104 Section 02
February 28, 2024

Administrative Notes

Interacting with the user

For this part of the class, we will generally get the input our programs need from the user typing the data in at the keyboard

- To get data, you prompt the user to type the data
- To get the ***right data***, you have to make sure that your prompt is clear and specific

Prompting

You prompt for the data using printf():

- Use printf() to tell the user what you need entered
- You can include literals and variables inside the parentheses
    - Literals are values between double quote characters “ “
    - Variables have to be specified
        - Type
        - Variable name
        - Width of the field to use

Reading

The basic way to read something in is scanf()

The basic syntax is

```
scanf("%(d or f or c or s)", &var_name)
```

- (d or f or c or s) means that you will put exactly one of those values there:
    - d if you want to read an integer
    - f if you want to read a floating point number
    - c if you want to read in a single character
    - s if you want to read in a string of one or more characters
- var_name is the name of the variable you want to have store the value you read in
    - Do NOT forget the &, which tells C to store the value in the location in memory associated with that variable name

Some rules to remember

- scanf() reads in the values until it gets to the end of a legitimate value of the type it's looking for
    - If %d - it's looking for an integer - it will stop reading at the first thing that isn't part of a valid integer - e.g., a blank space, a new line, a letter, a decimal point
    - If %f - it's looking for a floating point - it will stop reading at the first thing that isn't a valid part of a floating point number
    - If %c - it will read exactly one character, no matter what it is - even if it's a blank space or newline
    - If %s - it will read up to the specified number of characters, but it will stop at the first white space - blank space, tab or newline
- The next thing to be read will START right at that exact place where you left off - even on the same line

Some code we’ll walk through

```
#include <stdio.h>
int main () {
  int num;

  printf("Enter a number\n");
  scanf("%d", &num);
  printf("Number is %7d\n\n",num);

  float fnum;
  printf("Enter a floating point number\n");
  scanf("%f", &fnum);
  printf("Floating point is %1.4f\n\n", fnum);
```

```
char c;
  printf("Enter a character");
  scanf("%c",&c);
  printf("\nCharacter is %c\n\n", c);
  char str[20];
  printf("Enter a string");
  scanf("%20s", &str);
  printf("\nString is %20s\n\n", str);
  char s[20];
  printf("Enter another string\n");
  fgets(s, 20, stdin);
  printf("This string is %s \n\n",s);
}
```