

# Streams

Last updated 6/5/24

These slides introduce streams in C

# Streams

- Streams
  - Information flow between entities is done with **streams**
  - Keyboard → **Text input stream** → program data
    - **stdin**
  - Program data → **Text output stream** → Monitor
    - **stdout**
    - **stderr**                      // error messages
  - printf – formats data for the **text output stream**
  - scanf – formats data from the **text input stream**

**#include <stdio.h>**

# Streams

- `printf`
  - Combines text and data and inserts it into the **output stream**
  - text and data conversion is contained in double quotes
  - data is comma separated
  - data conversion is identified as `%xxx`

`% [flag] [min width] [precision] [size] code`

```
foo = 12.34567L;           // L indicates a long float
printf("%+6.3Lf", foo);
```

`%+6.3Lf` → `+12.345`    **sign, 6 total, 3 fractional, Long float**

# Streams

- printf

printf("%d%c%f", 12, 'a', 5.3); → 12a5.300000

no spaces

printf("%d %c %f", 12, 'a', 5.3); → 12 a 5.300000

spaces

```
int z;
```

```
z = 51;
```

```
printf("%d %f %c %x", z, z, z, z); → 51 51.00000 3 33
```

```
printf(" the value of z is: %d", z); → the value of z is: 51
```

```
printf{"I think %d is the value of z", z); → I think 51 is the value of z
```

Special characters are preceded by \

\n – new line, \t – tab, \% -%, \" – “

```
printf{"I think \"%d\" is the value of z", z); → I think "51" is the value of z
```



# Streams

- scanf
  - Extracts data from an **input stream** and formats it
    - lots of options
  - requires a **pointer** (addresses) for any variables
    - The pointer is required because scanf can read multiple values
  - text and data conversion is contained in double quotes
  - variable pointers are comma separated
  - whitespace is ignored – **except for characters**
  - data conversion is identified as %xxx

% [flag] [max width] [size] code

# Streams

- scanf

input      123 456 7a  
scanf("%d%d%d%c", &a, &b, &c, &d);  
a = 123, b=456, c=7, d='a'

xxx%c reads next character

input      123 456 7 a  
scanf("%d%d%d%c", &a, &b, &c, &d);  
a = 123, b=456, c=7, d=' '

xxx%c reads next character  
including whitespace

input      123 456 7 a  
scanf("%d%d%d %c", &a, &b, &c, &d);  
a = 123, b=456, c=7, d='a'

xxx %c ignores any whitespace  
(not just 1 space)  
then reads next character

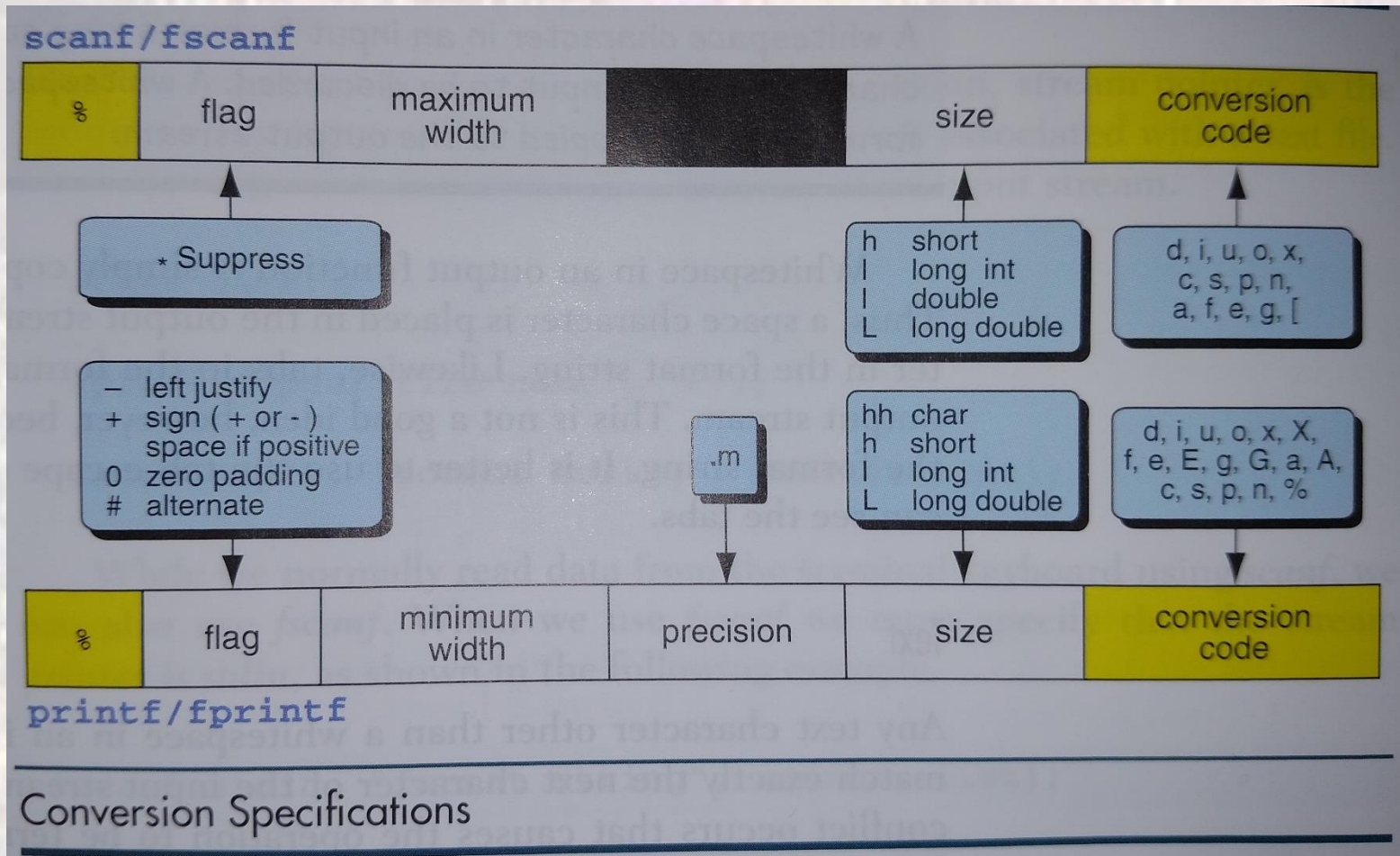
input      14/15 16/17  
scanf("%2d/%2d %2d/%2d", &num1, &den1, &num2, &den2);  
num1 = 14, den1 = 15, num2 = 16, den2 = 17

place characters you don't want  
to read in the format string

alternate approach: scanf("%2d%c%2d %2d%c%2d", &num1, &trash, &den1, &num2, &trash, &den2);

# Streams

- Stream I/O formatting



src: Forouzan



# Streams

- Stream I/O formatting

Argument Type	Size Specifier	Code
integral	hh (char), h (short), none (int), l (long), ll (long long)	i
integer	h (short), none (int), l (long), ll (long long)	d
unsigned int	hh (char), h (short), none (int), l (long), ll (long long)	u
character octal	hh (unsigned char)	o
integer hexadecimal	h (short), none (int), l (long), ll (long long)	x
real	none (float), l (double), L (long double)	f
real (scientific)	none (float), l (double), L (long double)	e
real (scientific)	none (float), l (double), L (long double)	g
real (hexadecimal)	none (float), l (double), L (long double)	a
character	none (char), l (wchar_t)	c
string	none (char string), l (wchar_t string)	s
pointer		p
integer (for count)	none (int), hh (char), h (short), l (long), ll (long long)	n
set	none (char), l (wchar_t)	[

**G  
A**