APS105
Winter 2012

Jonathan Deber
jdeber -at- cs -dot- toronto -dot- edu

Lecture 5
January 25, 2012

Today

- Reading Input
- More Conversion Specifiers

Constants and Style

```
int numCopies = 59;
```
⟵ Magic Number

Bad Style

```
#define FOUR 4
#define TWO_POINT_ONE 2.1
```

Bad Style

```
#define THIRTY 31
```

Evil

Style

- Two audiences:
  - Compiler
  - Other people (includes yourself in the future)

```
#include <ncurses.h>/*************************************************/
            int             m[256                   ] [         256   ],a
 ,b   ;;;   ;;;   WINDOW*w;   char*l=""   "\176qxl"   "q"   "q"   "k"   "w\
xm"   "x"   "t"         "j"         "v"         "u"         "n"         ,Q[
 ]=   "Z"   "pt!ftd`"   "qdc!`eu"   "dq!$c!nnwf"/**   ***   */"t\040\t";c(
int   u ,         int         v){                   v?m   [u]         [v-
 1]   |=2,m[u][v-1] &   48?W][v-1   ] &   15]]):0:0;u?m[u   -1][v]|=1   ,m[
 u-              1][   v]&         48?               W-1   ][v         ]&
15]   ]):0:0;v<   255   ?m[   u][v+1]|=8,m[u][v+1]&   48?   W][   v+1]&15]]
):0         :0;         u <               255   ?m[   u+1         ][v   ]|=
4,m[u+1][   v]&48?W+1][v]&15]]):0:0;W][   v]&   15]   ]);}cu(char*q){   return
 *q             ?cu   (q+         1)&         1?q   [0]               ++:
q[0   ]--   :1;   }d(   int   u ,   int/**/v,   int/**/x,   int   y){   int
Y=y   -v,   X=x         -u;   int         S,s   ;Y<         0?Y   =-Y   ,s,
s=-   1:(   s=1);X<0?X=-X,S   =-1  :(S=   1);   Y<<=   1;X<<=1;   if(X>Y){
int   f=Y               -(X   >>1   );;               while(u!=         x){
f>=   0?v+=s,f-=X:0;u   +=S   ;f+=   Y;m[u][v]|=32;mvwaddch(w,v   ,u,   m[u
 ][             v]&   64?   60:         46)         ;if         (m[   u][
v]&16){c(u,v);;   ;;;   ;;;   return;}}   }else{int   f=X   -(Y>>1);;   while
 (v   !=y         ){f   >=0         ?u   +=S,               f-=         Y:0
 ;v   +=s   ;f+=X;m[u][v]|=   32;mvwaddch(w,v   ,u,m[u][v]&64?60:46);if(m[u
 ][                  v]&         16)   {c(   u,v                     );
  ;   return;;;}}}}Z(   int/**/a,   int   b){   }e(   int/**/y,int/**/  x){
int             i ;         for         (i=         a;i               <=a
+S;i++)d(y,x,i,b),d(y,x,i,b+L);for(i=b;i<=b+L;i++)d(y,x,a,i),d(y,x,a+   S,i
 );                 ;;;         ;;;         ;;;               ;;;   ;
  mvwaddch(w,x,y,64);   ;;;   ;;;   ;;;   prefresh(   w,b,a,0,0   ,L-   1,S-1
);}            main(         int               V ,   char            *C[
  ]   ){FILE*f=   fopen(V==1?"arachnid.c"/**/   :C[   1],"r");int/**/x,y,c,
v=0         ;;;      initscr             ();               Z(Z         (raw
 ()   ,Z(   curs_set(0),Z(1   ,noecho()))),keypad(    stdscr,TRUE));w   =newpad
  (   300,  300             ) ;   for         (x=   255   ; x   >=0   ;x--
  )   for   (y=   255   ;y>=0;y--   )m[   x][   y]=   0;x=y=0;refresh( );while
arachnid by Nick Johnson, IOCCC 2004                  fgetc (f)   )+1)                    {if(
0||c==10||  x==   256){x=0;y++;if(y==256  )break;;}   else{m[x][y]=(c   ==
```

```
do {
    v++;
    mvwaddch (w, y,x ,m[x][ y]& 32? m[x ][y ] & 16? 0|
                          acs_map[l[m[x][y]&15]]:46:32);
    c==0163&&!(m[x][y+1]&16)?y++: 0;
    c == 119 &&! (m[ x][ y- 1]& 16) ?y--:0;
    ;
    c ==97 &&!(m[x-1][y]&16)?x--:0;
    c==100&&!(m[x+1 ][ y]& 16) ? x ++:0 ;
    if( c== 3- 1+1 ) {
        endwin( );
        ;
        return(0) ;
    }
    x -a<5?a>S- 5?a-=S-5:(a=0):0;
    x -a> S-5?a<255 -S* 2?a +=S-5:(a=256-S):0;
    y-b<5?b>L-5?b-=L-5:(b =0) :0;
    y-b>L-5?b<255-L *2?b+= L-5 :(b =256-L) :0;
    e(x,y);
    if(m[x][y]&64)break;
} while((c=getch())!=-1);
```

Good Variable Names

- Be clear (and use conventions)
- Don't abbreviate too much
  - `ns` vs. `numStudents`
- Don't be too verbose
  - `numberOfStudentsThatAreInTheClassAtThisTime`
- Use an initial comment

```
int numStudents;  // Number of students enrolled in class.
```

- It takes practice

Reading Input

scanf

- Reads "formatted" input

```
int mass = 18;
printf("%d", mass);

scanf("%d", &mass);
```

- Takes input, interprets and stores it
- Usually paired with `printf` prompt

Multiple Variables

- Multiple variables separated by whitespace

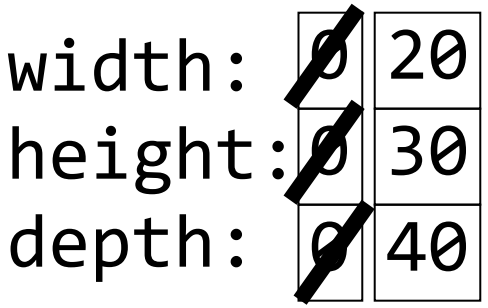
```
int width = 0, height = 0, depth = 0;
scanf("%d%d%d", &width, &height, &depth);
```

20 30 40

20

30

40



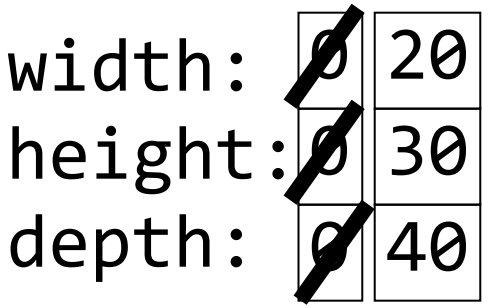
Multiple Variables

- Multiple variables separated by whitespace

```
int width = 0, height = 0, depth = 0;
scanf("%d%d%d", &width, &height, &depth);
```

```
20 30 40
```

```
20
    30
40
```

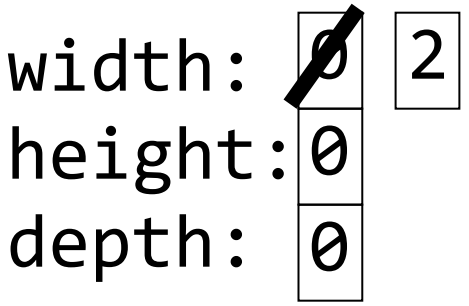



```
20␣30␣40↵
```

```
20␣␣␣␣␣␣␣␣␣30↵
␣␣␣␣␣␣40↵
```

Invalid Input

```
int width = 0, height = 0, depth = 0;
scanf("%d%d%d", &width, &height, &depth);
```



2 a 3 4

Invalid Input

```
int width = 0, height = 0, depth = 0;
scanf("%d%d%d", &width, &height, &depth);
int x = 0, y = 0;
scanf("%d%d", &x, &y);
```

2 a 3 4

'2'	' '	'a'	' '	'3'	' '	'4'	'↵'

width:	0
height:	0
depth:	0

Invalid Input

```
int width = 0, height = 0, depth = 0;
scanf("%d%d%d", &width, &height, &depth);
int x = 0, y = 0;
scanf("%d%d", &x, &y);
```

2 a 3 4

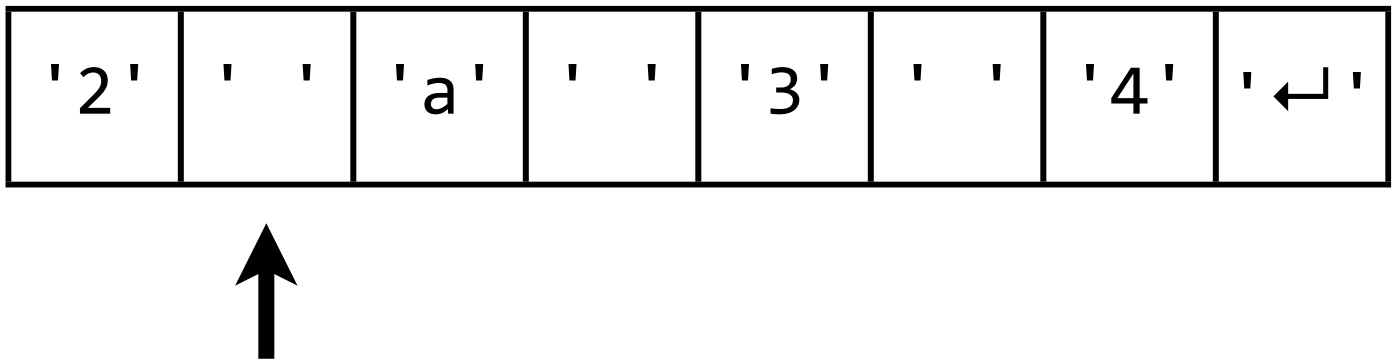
'2'	' '	'a'	' '	'3'	' '	'4'	'↵'

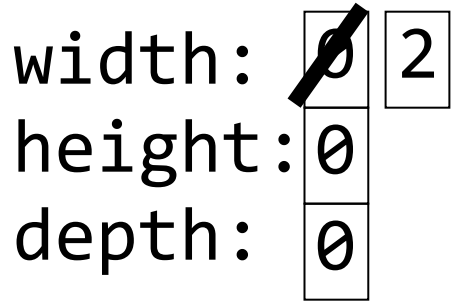
width:	0
height:	0
depth:	0

Invalid Input

```
int width = 0, height = 0, depth = 0;
scanf("%d%d%d", &width, &height, &depth);
int x = 0, y = 0;
scanf("%d%d", &x, &y);
```

2 a 3 4

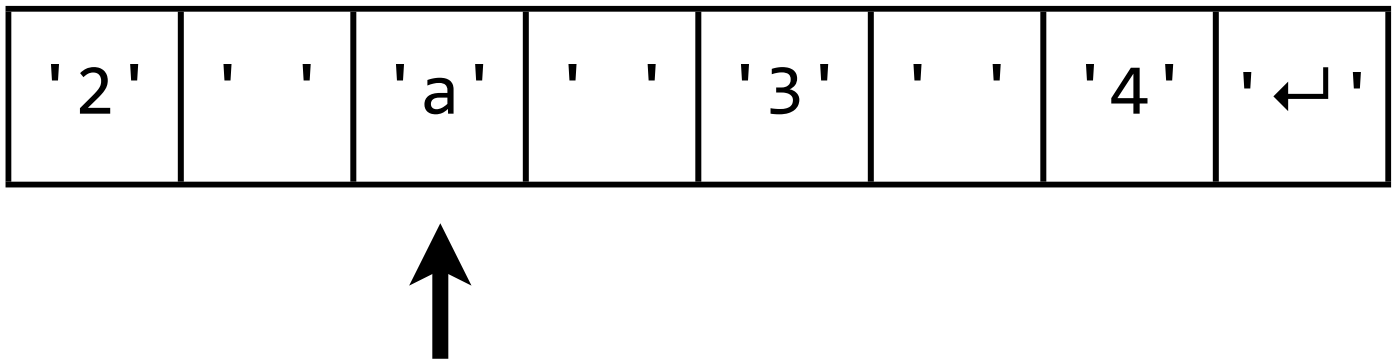


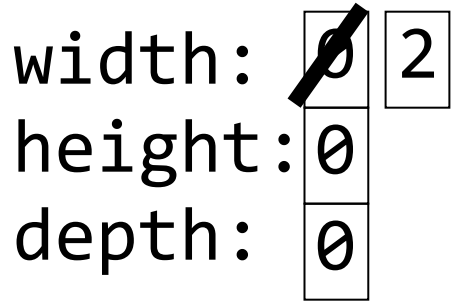


Invalid Input

```
int width = 0, height = 0, depth = 0;
scanf("%d%d%d", &width, &height, &depth);
int x = 0, y = 0;
scanf("%d%d", &x, &y);
```

2 a 3 4

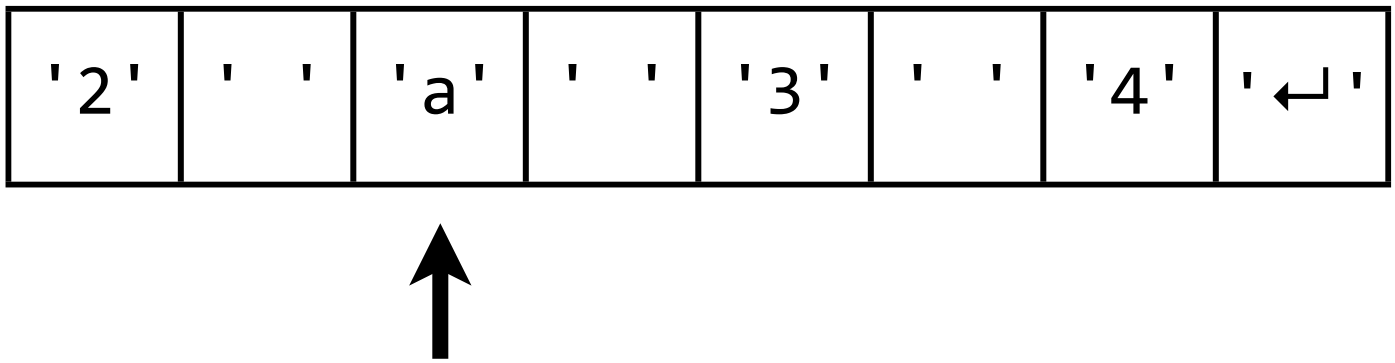


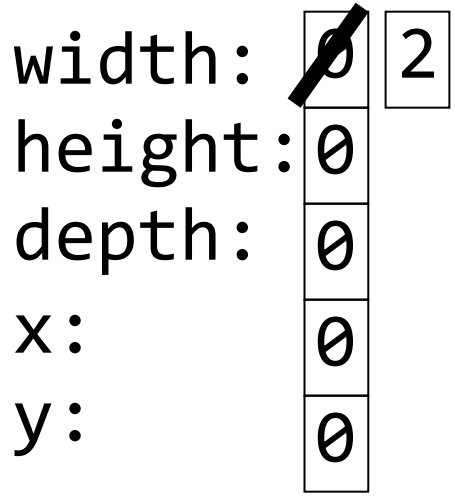


Invalid Input

```
int width = 0, height = 0, depth = 0;
scanf("%d%d%d", &width, &height, &depth);
int x = 0, y = 0;
scanf("%d%d", &x, &y);
```

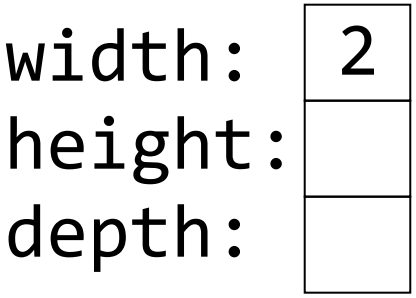
2 a 3 4





Invalid Input

```
int width, height, depth;
scanf("%d%d%d", &width, &height, &depth);
```



2 a 3 4

Format Strings

- Other characters can be present

```
int first = 0, second = 0;
scanf("%d-%d", &first, &second);
```

Format Specifiers

- `int`
  - `%d`
- `double`
  - `%lf` (different than `printf`!)
- `char`
  - `%c`
  - Doesn't skip whitespace

chars and scanf

- scanf does not skip whitespace for %c

```
int width = 0, height = 0, depth = 0;
scanf("%d%d%d", &width, &height, &depth);
```

20␣30␣40↵

20␣␣␣␣␣␣␣␣␣30↵

␣␣␣␣␣␣␣40↵

width:	20
height:	30
depth:	40

```
char first, middle, last;
scanf("%c%c%c", &first, &middle, &last);
```

JAD

first:	J
middle:	A
last:	D

chars and scanf

- scanf does not skip whitespace for %c

```
int width = 0, height = 0, depth = 0;
scanf("%d%d%d", &width, &height, &depth);
```

20␣30␣40↵

20␣␣␣␣␣␣␣␣␣␣30↵

␣␣␣␣␣␣␣␣40↵

width:	20
height:	30
depth:	40

```
char first, middle, last;
scanf("%c%c%c", &first, &middle, &last);
```

JAD

J␣␣␣␣␣␣␣␣␣A↵

␣␣␣␣␣D↵

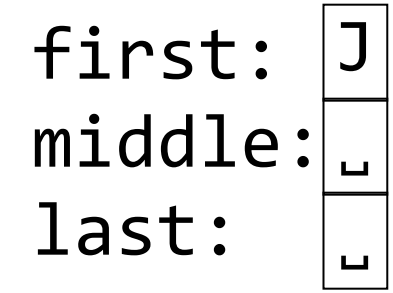
first:	J
middle:	␣
last:	␣

Skipping Whitespace

```
char first, middle, last;
scanf("%c%c%c", &first, &middle, &last);
```

J␣␣␣␣␣␣␣␣A↵

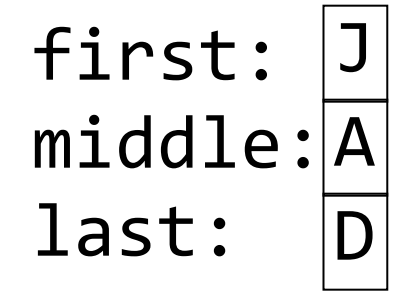
␣␣␣␣␣D↵




```
char first, middle, last;
scanf("%c %c %c", &first, &middle, &last);
```

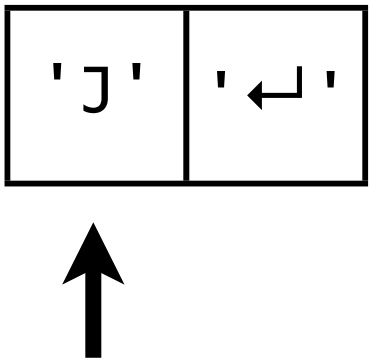
J␣␣␣␣␣␣␣␣A↵

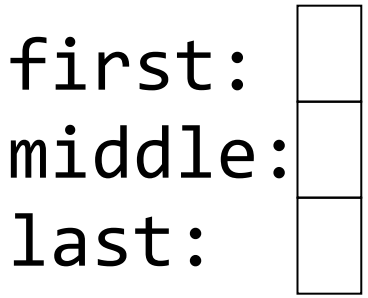
␣␣␣␣␣D↵



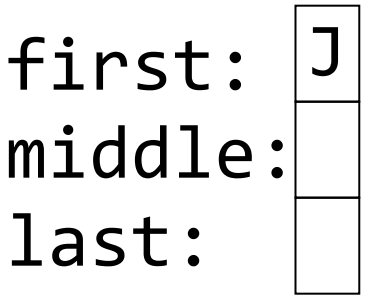
```
char first, middle, last;

printf("Please enter first initial: ");
scanf("%c", &first);
printf("Please enter middle initial: ");
scanf("%c", &middle);
printf("Please enter last initial: ");
scanf("%c", &last);
```

J ↵

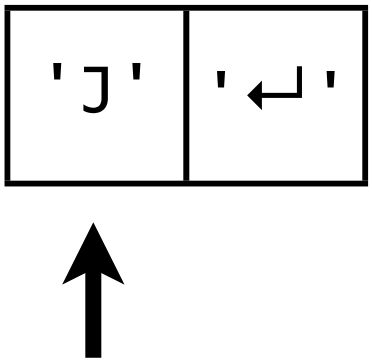




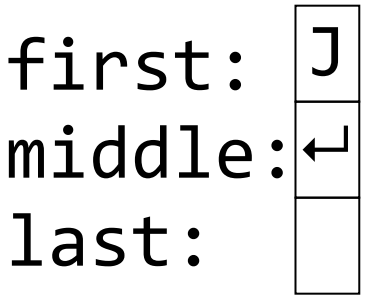
```
char first, middle, last;

printf("Please enter first initial: ");
scanf("%c", &first);
printf("Please enter middle initial: ");
scanf("%c", &middle);
printf("Please enter last initial: ");
scanf("%c", &last);
```



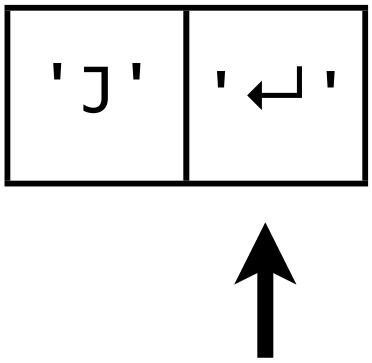
J ↵



```
char first, middle, last;

printf("Please enter first initial: ");
scanf("%c", &first);
printf("Please enter middle initial: ");
scanf("%c", &middle);
printf("Please enter last initial: ");
scanf("%c", &last);
```



J ↵

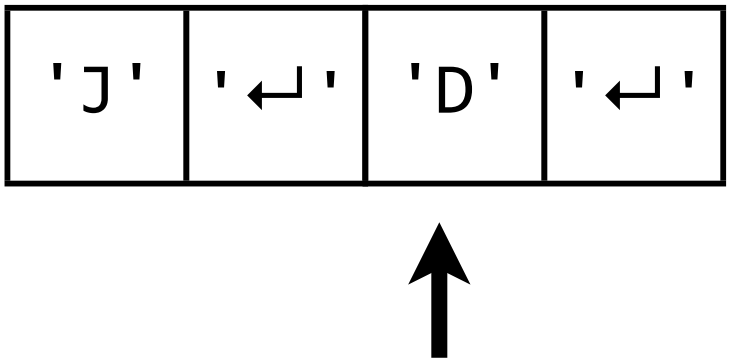


```
char first, middle, last;

printf("Please enter first initial: ");
scanf("%c", &first);
printf("Please enter middle initial: ");
scanf("%c", &middle);
printf("Please enter last initial: ");
scanf("%c", &last);
```

J ↵

```
char first, middle, last;

printf("Please enter first initial: ");
scanf("%c", &first);
printf("Please enter middle initial: ");
scanf("%c", &middle);
printf("Please enter last initial: ");
scanf("%c", &last);
```

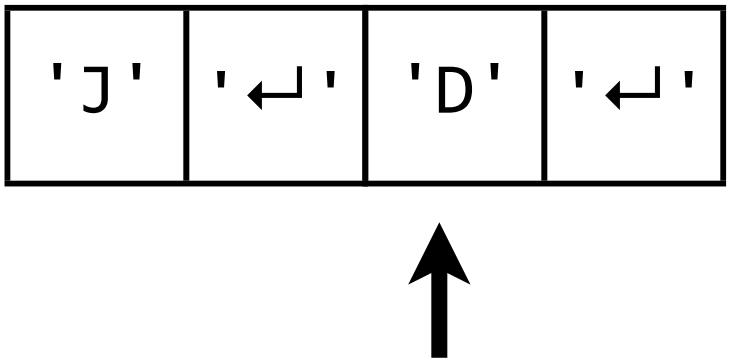
J ↵ D ↵



```
char first, middle, last;

printf("Please enter first initial: ");
scanf("%c", &first);
printf("Please enter middle initial: ");
scanf("%c", &middle);
printf("Please enter last initial: ");
scanf("%c", &last);
```

J ↵ D ↵

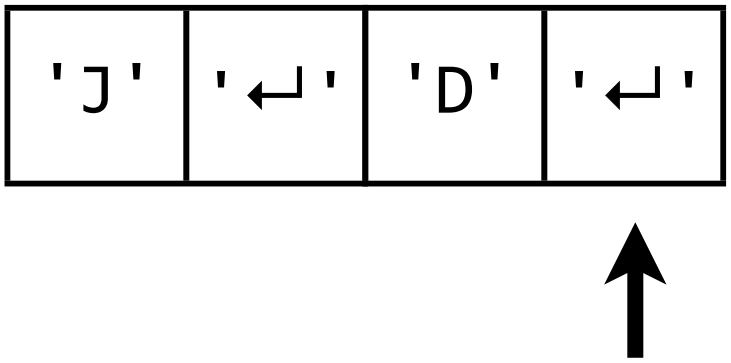
first:	J
middle:	↵
last:	D



```
char first, middle, last;

printf("Please enter first initial: ");
scanf("%c", &first);
printf("Please enter middle initial: ");
scanf("%c", &middle);
printf("Please enter last initial: ");
scanf("%c", &last);
```

J ↵ D ↵

first:	J
middle:	↵
last:	D



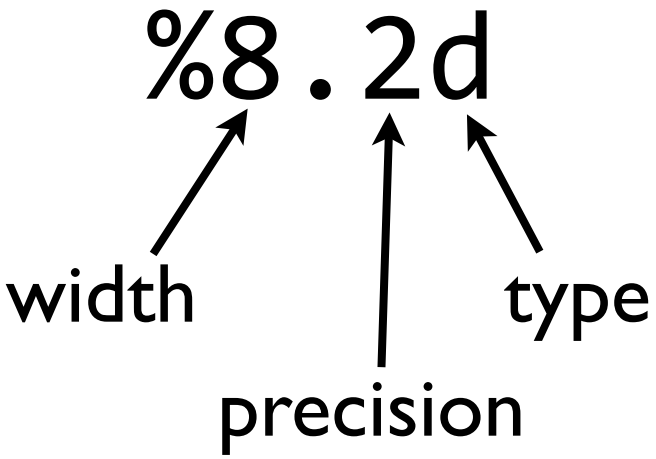
scanf

- Is a bit awkward
- Not really designed for human use

- We'll see some better alternatives later

Customizing `printf` Conversion Specifiers

```
int i = 3;
printf("|%d| \n", i);          |3|
printf("|%8.2d| \n", i);       |      03|
printf("|%8d| \n", i);         |       3|
printf("|%.2d| \n", i);        |03|

double d = 4.567;
printf("|%g| \n", d);          |4.567|
printf("|%8.2g| \n", d);       |     4.6|
printf("|%8g| \n", d);         |   4.567|
printf("|%.2g| \n", d);        |4.6|
```


%8.2d
width
precision
type

Width

- Same meaning for all conversion specifiers
- Indicates the *minimum* number of characters to use

```
int i = 345;
printf("|%d| \n", i);      |345|
printf("|%8d| \n", i);     |␣␣␣␣␣345|
printf("|%2d| \n", i);     |345|

double d = 4.567;
printf("|%g| \n", d);      |4.567|
printf("|%8g| \n", d);     |␣␣␣4.567|
printf("|%2g| \n", d);     |4.567|
```

Precision

- Different specifiers interpret it differently
- %d
- %g
- %f

Precision (%d)

- For %d, it's the *minimum* number of digits

```
int i = 345;
printf("|%d| \n", i);      |345|
printf("|%.8d| \n", i);    |00000345|
printf("|%.2d| \n", i);    |345|
```

Precision (%g)

- For %g, it's the *maximum* number of significant digits

```
double d = 4.567;
printf("|%g| \n", d);     |4.567|
printf("|%.8g| \n", d);   |4.567|
printf("|%.2g| \n", d);   |4.6|

double e = 2.99792458e8;
printf("|%g| \n", e);     |2.99792e+08|
printf("|%.10g| \n", e);  |299792458|
printf("|%.4g| \n", e);   |2.998e+08|
```

Precision (%f)

- For %f, it's the *exact* number of digits after the decimal

```
double d = 4.567;
double e = 2.99792458e8;
```

By default, %f means %.6f

```
printf("|%f| \n", d);     |4.567000|
printf("|%f| \n", e);     |299792458.000000|

printf("|%.8f| \n", d);   |4.56700000|
printf("|%.8f| \n", e);   |299792458.00000000|

printf("|%.1f| \n", d);   |4.6|
printf("|%.1f| \n", e);   |299792458.0|
```