Basic I/O

- There is no input or output defined in C itself
- Character based (no format specifiers)
  - getchar() - input
  - putchar(c) - output
- Formatted
  - scanf(stuff goes in here) - input *** white space is important!!!
  - printf(stuff goes in here) - output
  - Format Specifiers (% before specifier) – see next slide

```
#include <stdio.h>
int main(void) {
    int i = 65; /* what if 258 instead of 65? */
    char a;
    printf("i=\n",i);
    printf("output with a putchar ");
    putchar(i);
    a = (char) i;
    printf("a=\n",a);
    getchar();
    return(0); } /* check.c */
```

```
#include <stdio.h>
int main() {    /* check1.c */
    int x;
    scanf("%d\n", &x);
    printf("x=%d\n", x); }
```

Q. Why are pointers given to scanf?
A. Needs a pointer to the variable if it is going to change the variable itself i.e. assign a value to x.
When programming in C, you use conversion characters — the percent sign and a letter, for the most part — as placeholders for variables you want to display. The following table shows the conversion characters and what they display:

<table>
<thead>
<tr>
<th>Conversion Character</th>
<th>Displays Argument (Variable’s Contents) As</th>
</tr>
</thead>
<tbody>
<tr>
<td>%c</td>
<td>Single character</td>
</tr>
<tr>
<td>%d</td>
<td>Signed decimal integer (int)</td>
</tr>
<tr>
<td>%e</td>
<td>Signed floating-point value in E notation</td>
</tr>
<tr>
<td>%f</td>
<td>Signed floating-point value (float)</td>
</tr>
<tr>
<td>%g</td>
<td>Signed value in %e or %f format, whichever is shorter</td>
</tr>
<tr>
<td>%i</td>
<td>Signed decimal integer (int)</td>
</tr>
<tr>
<td>%o</td>
<td>Unsigned octal (base 8) integer (int)</td>
</tr>
<tr>
<td>%s</td>
<td>String of text</td>
</tr>
<tr>
<td>%u</td>
<td>Unsigned decimal integer (int)</td>
</tr>
<tr>
<td>%x</td>
<td>Unsigned hexadecimal (base 16) integer (int)</td>
</tr>
<tr>
<td>%%</td>
<td>(percent character)</td>
</tr>
</tbody>
</table>
#include <stdio.h>
int main(void) {  
    int i = 65;
    /* what if 258 instead of 65? */
    char a;
    printf("i=%d\n",i);  
    printf("output with a putchar ");
    putchar(i);
    printf("\ni=%i",i);
    a = (char) i;
    printf("\na=%c\n",a);
    i=getchar();
    printf("i=%c\n",i);
    printf("i=0x%x\n",i);
    printf("i=%d\n",i);
    return (0);
}

#include <stdio.h>
#define PI 3.14159265358979323846
int main() { 
    int x;
    scanf("%d", &x); /* why need & ? */
    printf("%d\n", x);
    float var;
    scanf("%f", &var);
    scanf("%d", &var);
    scanf("%lf", &var);
    scanf("enter value ", &var);
    int first, second;
    scanf("%d%d", &first, &second);
    scanf(" %d %*d %*d%*d %d ", &i, &j)
    return 0;  }
# Printf formatted output conversions

<table>
<thead>
<tr>
<th>Character</th>
<th>Argument type; Printed As</th>
</tr>
</thead>
<tbody>
<tr>
<td>d,i</td>
<td>int; signed decimal notation.</td>
</tr>
<tr>
<td>o</td>
<td>int; unsigned octal notation (without a leading zero).</td>
</tr>
<tr>
<td>x,X</td>
<td>unsigned int; unsigned hexadecimal notation (without a leading 0x or 0X), using abcdef for 0x or ABCDEF for 0X.</td>
</tr>
<tr>
<td>u</td>
<td>int; unsigned decimal notation.</td>
</tr>
<tr>
<td>c</td>
<td>int; single character, after conversion to unsigned char</td>
</tr>
<tr>
<td>s</td>
<td>characters from the string are printed until a ‘\0’ is reached or until the number of characters indicated by the precision have been printed.</td>
</tr>
<tr>
<td>f</td>
<td>double; decimal notation of the form [-]mmm.ddd, where the number of d’s is given by the precision. The default precision is 6; a precision of 0 suppresses the decimal point.</td>
</tr>
<tr>
<td>e,E</td>
<td>double; decimal notation of the form [-]m.dddde+/-xx or [-]m.dddddE+/-xx, where the number of d’s is specified by the precision. The default precision is 6; a precision of 0 suppresses the decimal point.</td>
</tr>
<tr>
<td>g,G</td>
<td>double; %e or %E is used if the exponent is less than -4 or greater than or equal to the precision; otherwise %f is used. Trailing zeros and a trailing decimal point are not printed.</td>
</tr>
</tbody>
</table>
Decimal & Floating point

- **%d**: print as decimal integer
- **%6d**: print as decimal integer, at least 6 characters wide
- **%f**: print as floating point
- **%6f**: print as floating point, at least 6 characters wide
- **%.2f**: print as floating point, 2 characters after decimal point
- **%6.2f**: print as floating point, at least 6 wide and 2 after decimal point

- Width of the whole number portion decimal integer
Printf examples

causes the values of the two integers fahr and celsius to be printed, with a tab (\t) between them

```c
printf("%d\t%d\n", fahr, celsius);
```

to print the first number of each line in a field three digits wide, and the second in a field six digits wide

```c
printf("%3d %6d\n", fahr, celsius);
```

Each % construction in the first argument of printf is paired with the corresponding second argument, third argument, etc.; they must match up properly by number and type, or you will get wrong answers.

```c
printf("\na=%f\nb=%f\nc=%f\nPI=%f", a, b, c, d);
```

```c

void main()
{
    floatingpointno a;
    clrscr();
    if(TRUE)
    {
        a=PI;
        printf("a=%f\nPI=%f",a,PI);
    }
    getch();
}
```

c = a + b;
printf("%d + %d = %d\n", a, b, c);
Scanf formatted input conversions

- Consists of % at the beginning and a type indicator at the end
- In between options:
  - *= used to suppress input
  - maximum field-width indicator
  - type indicator modifier
- Scanf requires two inputs:
  - String argument
  - Set of additional arguments

<table>
<thead>
<tr>
<th>Character</th>
<th>Input Data; Argument type</th>
</tr>
</thead>
<tbody>
<tr>
<td>d</td>
<td>decimal integer</td>
</tr>
<tr>
<td>i</td>
<td>integer; the integer may be in octal (leading 0) or hexadec.pngi.pngal (leading 0x or 0X).</td>
</tr>
<tr>
<td>o</td>
<td>octal integer (with or without leading zero)</td>
</tr>
<tr>
<td>u</td>
<td>unsigned decimal integer;</td>
</tr>
<tr>
<td>x</td>
<td>hexadecimal integer (with or without leading 0x or 0X)</td>
</tr>
<tr>
<td>e,f,g</td>
<td>floating-point number; the input format for float’s is an optional sign, a string of numbers possibly containing a decimal point, and an optional exponent field containing an E or e followed by a possibly signed integer.</td>
</tr>
</tbody>
</table>