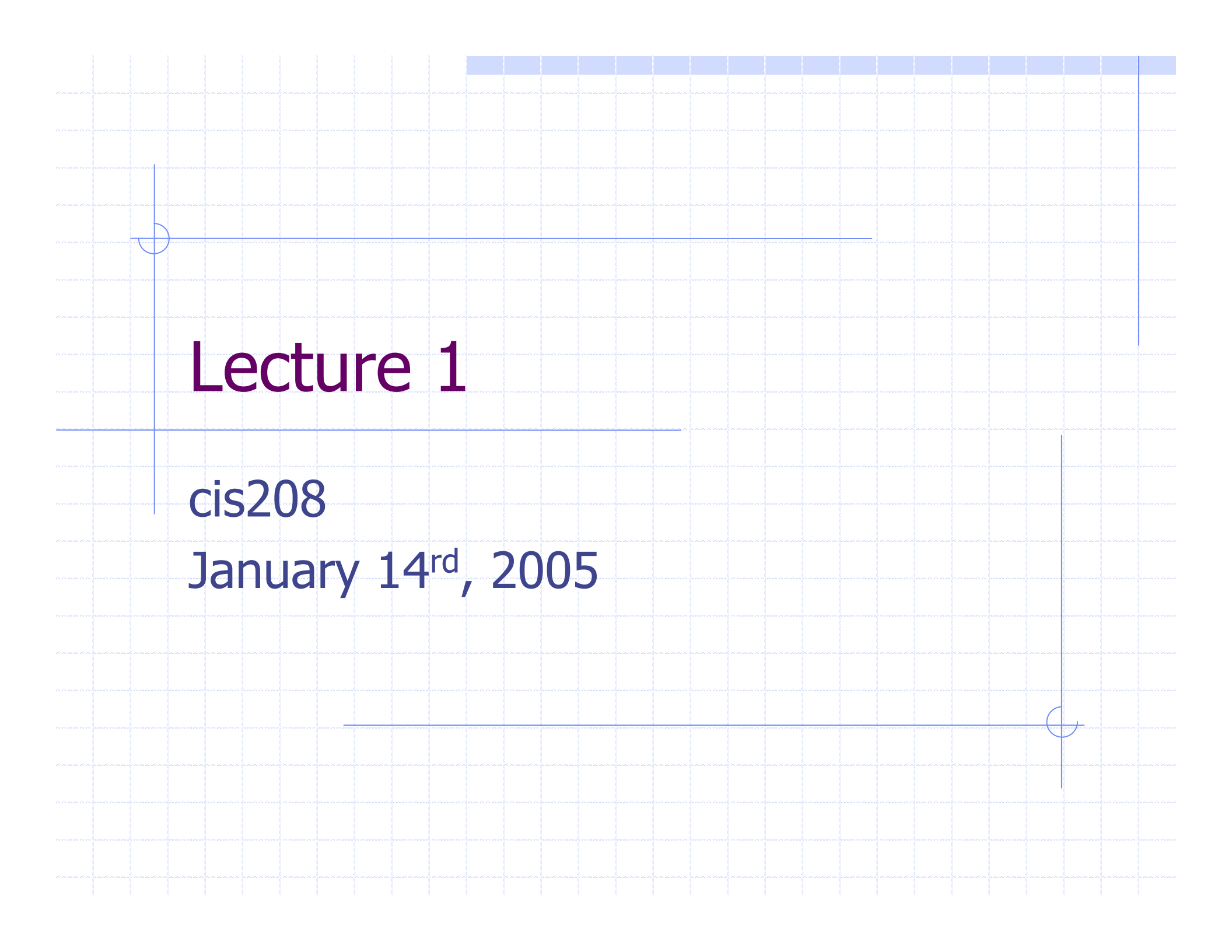




Lecture 1

cis208

January 14rd, 2005

Compiling

%> gcc helloworld.c
returns a.out

%> gcc -o helloworld helloworld.c
returns helloworld

Executing

%> a.out

%> ./a.out

%> helloworld

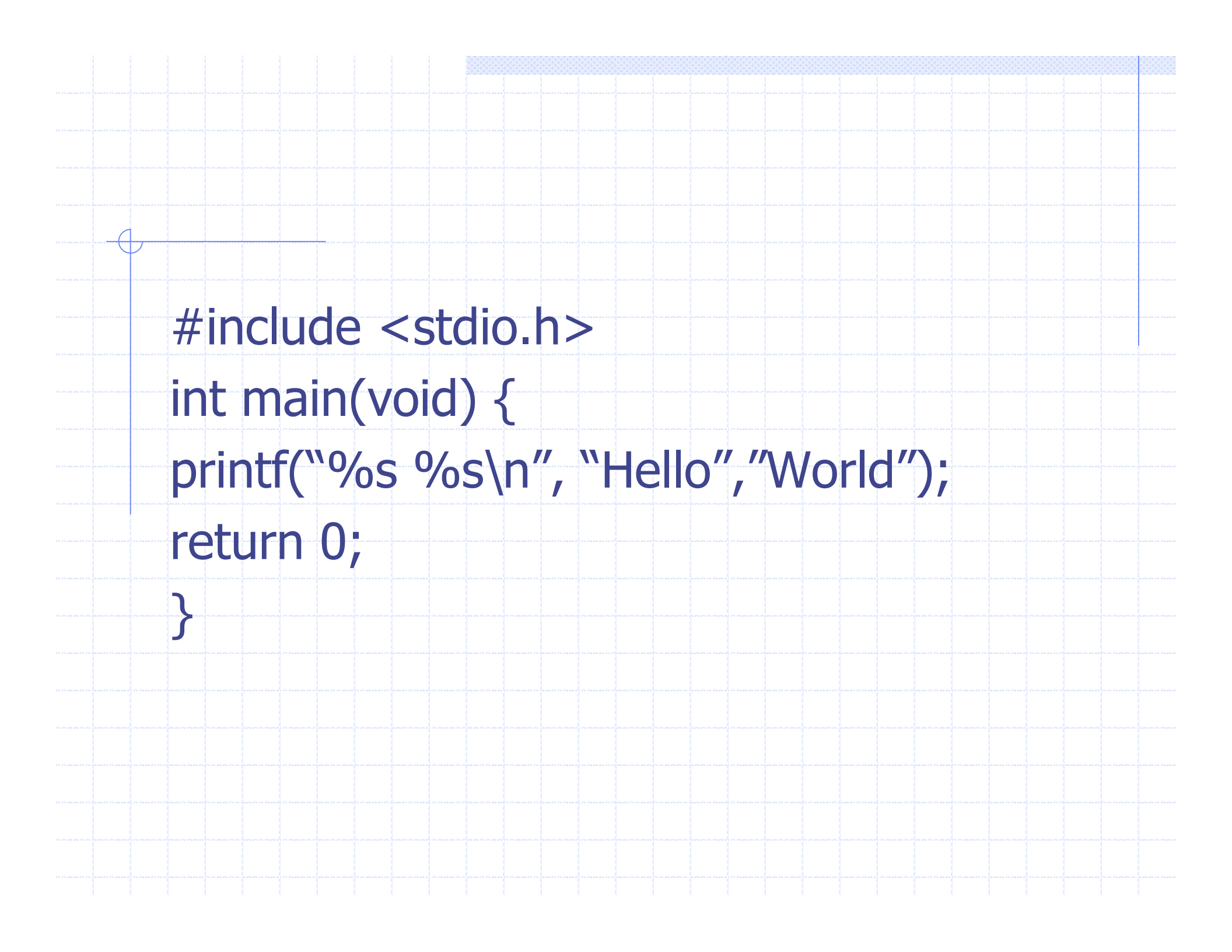
%> ./helloworld

The C Language

- ◆ Derived from B & BCPL
- ◆ Redefined as ANSI C
American National Standards Institute

C / Java Differences

- ◆ Not Object Orientated
- ◆ Procedural implementation and design
- ◆ No Polymorphism
- ◆ Call-by-Value only
- ◆ No Virtual Machine
- ◆ Direct Memory Access and allocation
- ◆ Very Fast
- ◆ Easier to Optimize.



```
#include <stdio.h>
int main(void) {
printf("%s %s\n", "Hello", "World");
return 0;
}
```

#include <stdio.h>

- ◆ Adds other source files
- ◆ Similar to java's import statements
- ◆ Implicit nesting - 8 Levels

#include continued

◆ Preprocessor command

#include <stdio.h>

<> : standard libraries.

#include "foo.h"

" " : local h files. Give path.

Common Libraries

◆ `stdio.h` : standard I/O

- ◆ `printf`, `scanf`, `gets`, `puts`, `getch`, etc

◆ `stdlib.h` : various functions

- ◆ `rand`, `abs`, `bsearch`, `exit`, etc

◆ `string.h` : string functions

- ◆ `strlen`, `memcpy`, `strcat`, `strchr`, etc

◆ `math.h` : math functions

- ◆ must compile with `-lm` option
- ◆ trig functions, `log`, `pow`, `sqrt`

int main(void)

- ◆ int is return type.

- ◆ must have main function

- ◆ in gcc, arguments & returns not necessary if void.

Variable Declaration

- ◆ Must be at beginning
- ◆ No initialization == junk values.
- ◆ = is assignment operator
 - variable = value;
 - value = variable;

Correct
Wrong

declaration cont.

```
int main(void) {  
    char a, b='b';  
    int j,i,k=8;  
    int p;  
    .  
    .  
}
```

basic types Page 110

- ◆ char : characters. 'a', 'b', '1', '2', etc
- ◆ int : integers. 1, 3 1000, -11, etc
- ◆ float : floating point decimals
 - ◆ 1.11, 0.002, 1.231, etc
- ◆ double : same as float, but larger range

printf

- ◆ part of `<stdio.h>`
- ◆ Formatted output to standard out
 - ◆ usually console
 - ◆ can be changed
 - ◆ advanced topic

printf

```
int printf(const char *control_string, ... , );
```

returns number of characters written to stdout

control_string (printf cont)

- ◆ Represents what will be printed.

```
printf("I like C \n");
```

- ◆ \n is newline character

```
printf("I\nlike\nC\n");
```

I

like

C

printf cont...

\ == escape character

\n : newline

\t : horizontal tab

page 113

format specifiers %

- ◆ variable place holder in control string
 - values plugged in during runtime

```
printf("%d\n%c %s\n", 1, 'a', "nice");
```

1

a nice

- ◆ 1-to-1 correspondence

format specifiers

◆ different types of input

- %c – character
- %d – decimal value
- %i - same as %d
- %f – float value
- %s – string

```
int main(void){  
    int i = 5, j = 2, k = 3;  
    char a = '7';  
    printf("%d + %d equals %c\n", j, i, a);  
    i = printf("%d", i+j+k);  
    return 0;  
}
```

Minimum Field width

- ◆ numerical padding
- ◆ numerical values between % and specifier
- ◆ adds white spaces or zeros

```
printf("%05d%5d",5,6);
```

```
00005    6
```

Mostly used to line up columns in console output

Precision Specifier

◆ sets number of significant digits

```
float j = 5.0;  
printf("%2.5f\n",j);
```

two white spaces and 7 digits after decimal point.