CPS 188

Computer Programming Fundamentals

Prof. Alex Ufkes

Topic 2.1: Overview of C, variables, types, output

Toronto Metropolitan University

Notice!

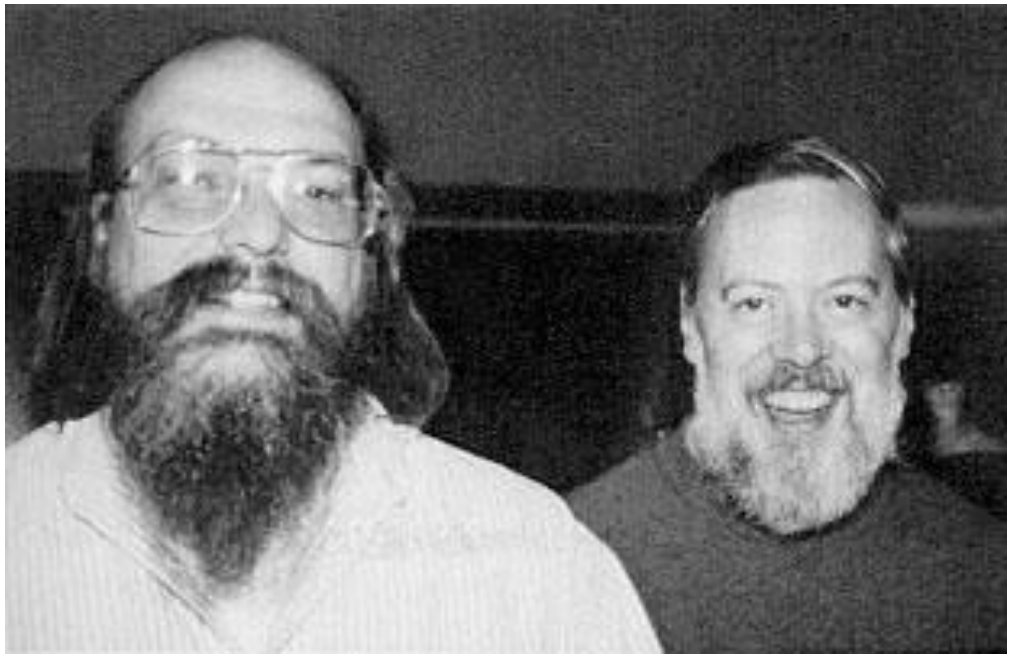
Obligatory copyright notice in the age of digital delivery and online classrooms:

Today

Intro to C

- Data types, variables
- Basic program structure
- Formatted input/output

What, Why, When, Who?



- Dennis Ritchie & Ken Thompson
- Bell Labs, 1972-1973
- Wanted to write utilities for Unix
- This was early Unix, being developed in Assembly at the time.
- Later, C was used to re-implement the entire Unix kernel.
- One of the first kernels implemented in something *other* than Assembly!

What, Why, When, Who?

C was one of the early ***general-purpose*** programming languages.

Meaning: it wasn't developed with a specific application domain in mind

FORTRAN – **FOR**mula **TRAN**slation

- Turn mathematical formulas into code

COBOL – **CO**mmon **B**usiness **O**riented **L**anguage

- Designed for business, finance use

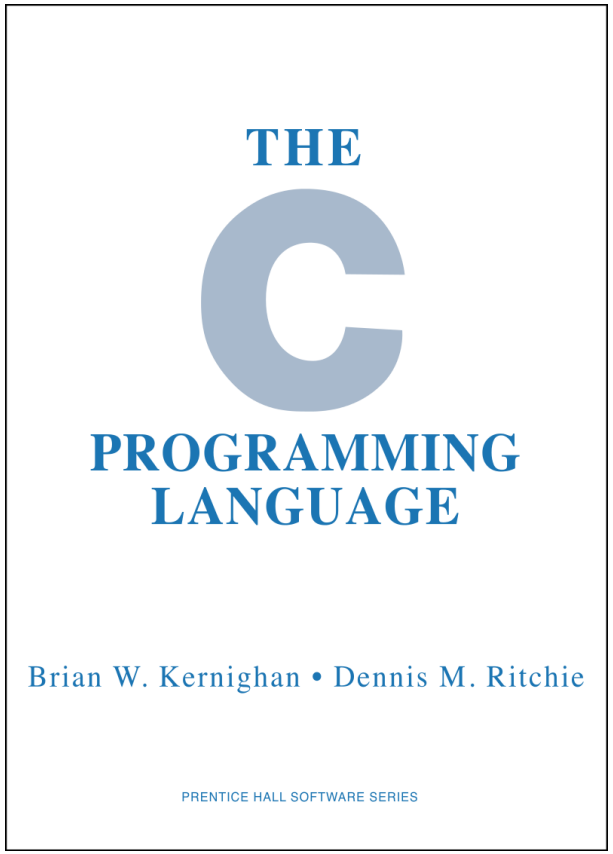
ALGOL – Meant for algorithm description

What, Why, When, Who?

Why teach C?

- C is relatively *small* (32 reserved words)
- C is *common*, it's everywhere
- C is *stable* (well established language, doesn't change much anymore)
- C is *efficient* at runtime
- C is the *basis for other languages* (e.g. C++)
- C is (relatively) *easy to learn*.

What, Why, When, Who?



The original C manual…

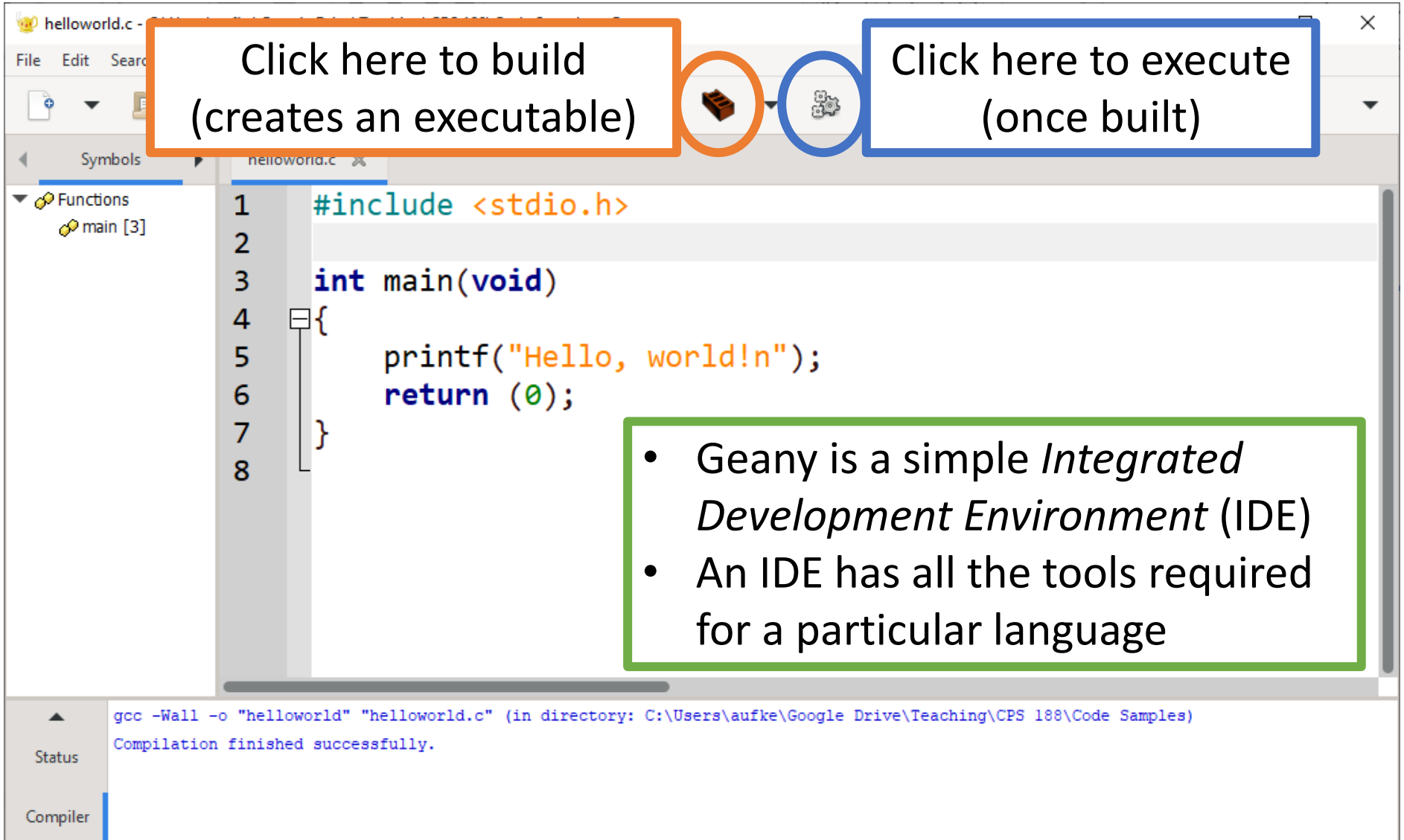
```
#include <stdio.h>

int main (void)
{
     printf ("Hello, world!\n");

     return (0);
}
```

…was the first to say hello to the world.

Hello, world!

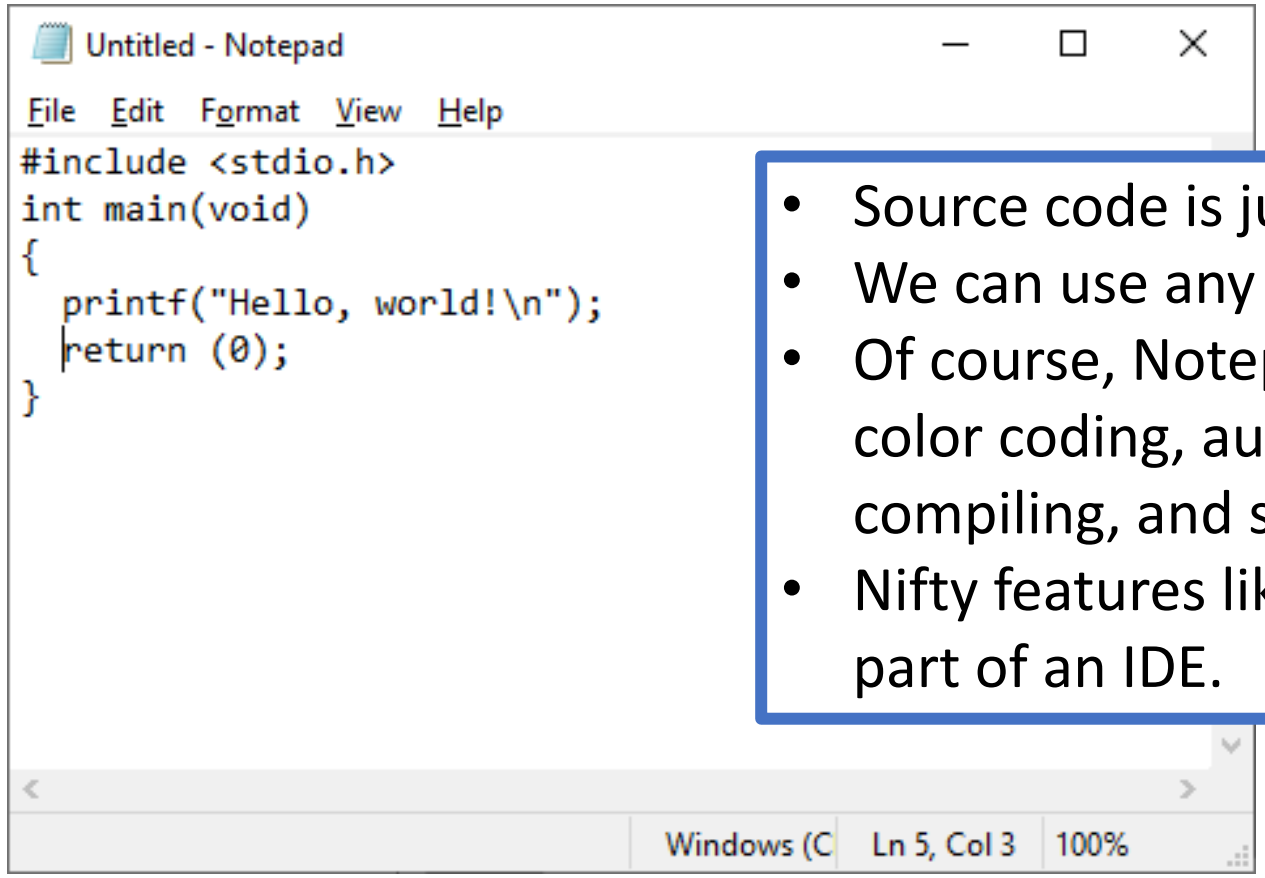
```
#include <stdio.h>
int main(void)
{
   printf("Hello, world!\n");
   return (0);
}
```

Development Tools: Editor



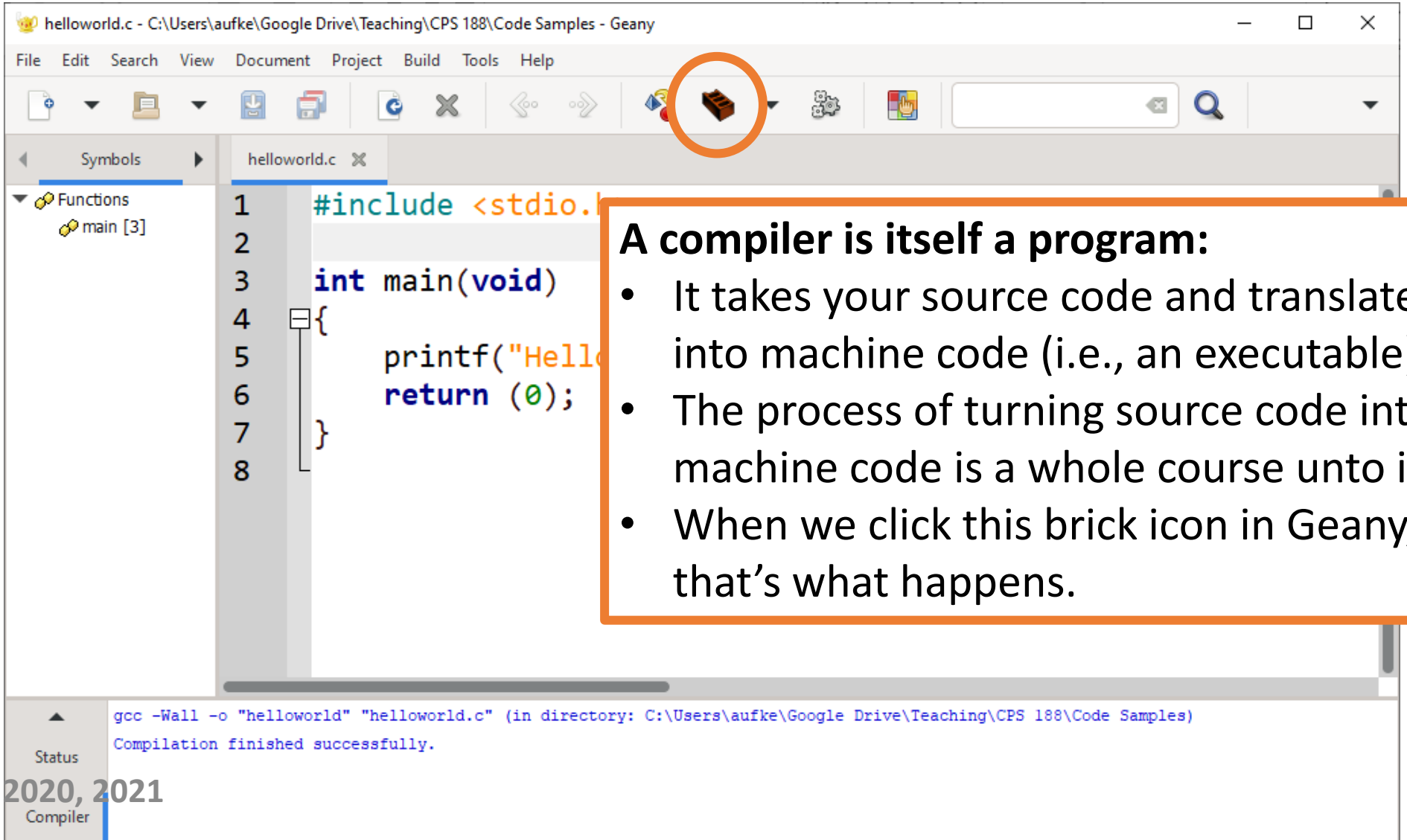
- Geany is a simple *Integrated Development Environment* (IDE)
- An IDE has all the tools required for a particular language

Development Tools: Editor



- Source code is just plain text.
- We can use any plain text editor.
- Of course, Notepad doesn't have color coding, autocomplete, 1-click compiling, and so on.
- Nifty features like these are typically part of an IDE.

Development Tools: Compiler



A compiler is itself a program:

- It takes your source code and translates it into machine code (i.e., an executable).
- The process of turning source code into machine code is a whole course unto itself
- When we click this brick icon in Geany, that's what happens.

Digging Deeper: Hello, world!

Pre-processor directive ?

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

function ***prototype*** or ***signature***

main function

Semi-colon terminates C statements

Curly braces indicate ***scope*** of main function

Compilation Stages

1.

```
#include <stdio.h>

int main (void)
{
     printf ("Hello, world!\n");

     return (0);
}
```

Source Code (C)

2.

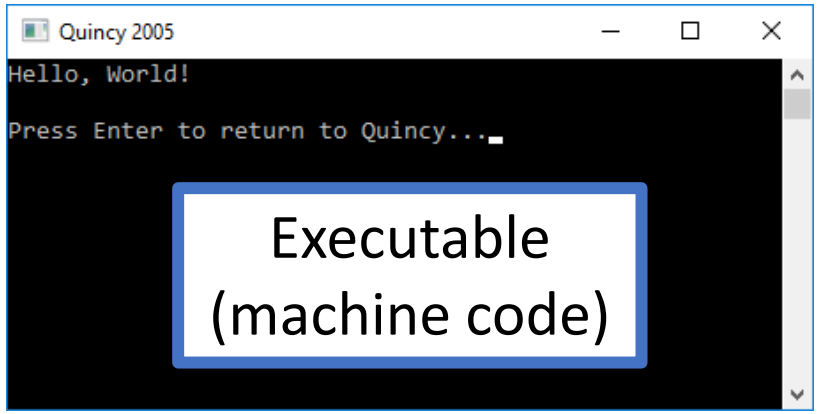
```
#include <stdio.h>

int main (void)
{
     printf ("Hello, world!\n");

     return (0);
}
```

Preprocessor

3.

Compiler

4.



Executable (machine code)

Preprocessor Directives

- The preprocessor modifies your C source code prior to compilation.
- Preprocessor directives begin with **#**
- There are many different directives, but in this course, we'll deal primarily with two:
- **`#include`** and **`#define`**

Preprocessor Directives: `#include`

```
#include <stdio.h>

int main(void)
{
   printf("Hello, world!\n");
   return (0);
}
```

- Tells the compiler to include the ***header file*** stdio.h
- stdio, or "standard input/output", is a ***library***.
- A library is a collection of functions, symbols, values
- Put simply, code that already exists that we can use

stdio.h

```
int  puts(const char *);
int  remove(const char *);
int  rename (const char *, const char *);
void        rewind(FILE *);
int  scanf(const char * __restrict, ...) __scanflike(1, 2);
void        setbuf(FILE * __restrict, char * __restrict);
int  setvbuf(FILE * __restrict, char * __restrict, int, size_t);
int  sprintf(char * __restrict, const char * __restrict, ...) __printflike(2, 3);
int  sscanf(const char * __restrict, const char * __restrict, ...) __scanflike(2, 3);
FILE       *tmpfile(void);

#if !defined(_POSIX_C_SOURCE)
__deprecated_msg("This function is provided for compatibility reasons only.  Due to security concerns inherent in the design o
it is highly recommended that you use mkstemp(3) instead.")
#endif
char       *tmpnam(char *);
int  ungetc(int, FILE *);
int  vfprintf(FILE * __restrict, const char * __restrict, va_list) __printflike(2, 0);
int  vprintf(const char * __restrict, va_list) __printflike(1, 0);
int  vsprintf(char * __restrict, const char * __restrict, va_list) __printflike(2, 0);
__END_DECLS



/* Additional functionality provided by:
 * POSIX.1-1988
 */

#if __DARWIN_C_LEVEL >= 198808L
#define    L_ctermid  1024 /* size for ctermid(); PATH_MAX */

__BEGIN_DECLS
#ifndef __CTERMID_DEFINED
/* Multiply defined in stdio.h and unistd.h by SUS */
#define __CTERMID_DEFINED 1
char       *ctermid(char *);
#endif

#if defined(_DARWIN_UNLIMITED_STREAMS) || defined(_DARWIN_C_SOURCE)
FILE       *fdopen(int, const char *) __DARWIN_ALIAS_STARTING(__MAC_10_6, __IPHONE_3_2, __DARWIN_EXTSN(fdopen));
#else /* !_DARWIN_UNLIMITED_STREAMS && !_DARWIN_C_SOURCE */
FILE       *fdopen(int, const char *) __DARWIN_ALIAS_STARTING(__MAC_10_6, __IPHONE_2_0, __DARWIN_ALIAS(fdopen));
#endif /* (DARWIN_UNLIMITED_STREAMS || _DARWIN_C_SOURCE) */
int  fileno(FILE *);
__END_DECLS
#endif /* __DARWIN_C_LEVEL >= 198808L */


/* Additional functionality provided by:
 * POSIX.2-1992 C Language Binding Option
 */

#if __DARWIN_C_LEVEL >= 199209L
__BEGIN_DECLS
int  pclose(FILE *);
#if defined(_DARWIN_UNLIMITED_STREAMS) || defined(_DARWIN_C_SOURCE)
FILE       *popen(const char *, const char *) __DARWIN_ALIAS_STARTING(__MAC_10_6, __IPHONE_3_2, __DARWIN_EXTSN(popen));
#else /* !_DARWIN_UNLIMITED_STREAMS && !_DARWIN_C_SOURCE */
FILE       *popen(const char *, const char *) __DARWIN_ALIAS_STARTING(__MAC_10_6, __IPHONE_2_0, __DARWIN_ALIAS(popen));
```

Preprocessor Directives: `#include`

```
#include <stdio.h>

int main(void)
{
  printf("Hello, world!\n");
  return (0);
}
```

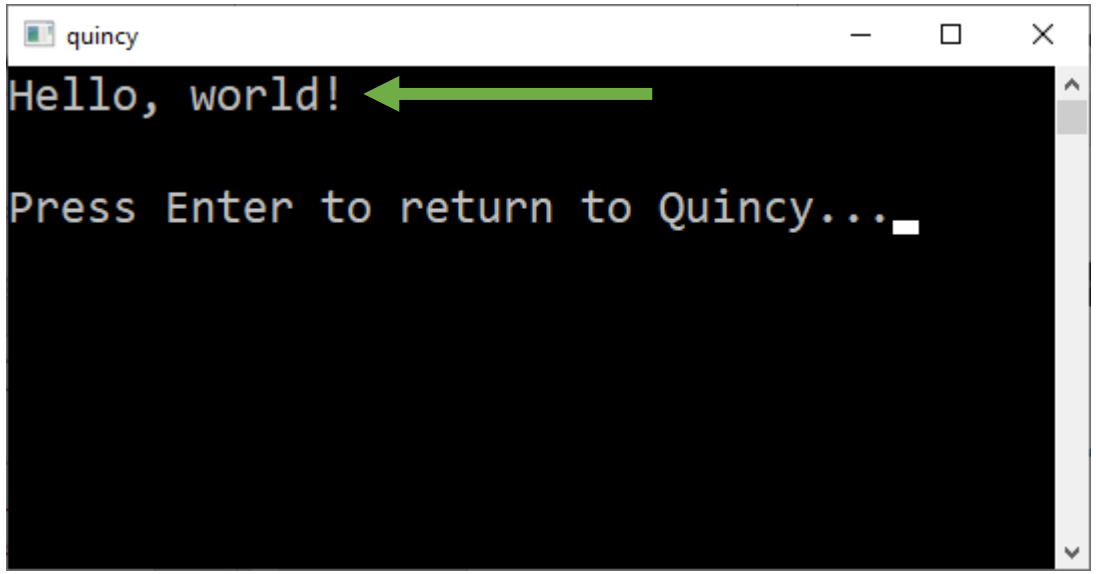
What does `stdio` get us?

- Many things! one of which is access to the **`printf`** function

Preprocessor Directives: #define

#define acts as a search-and-replace macro:

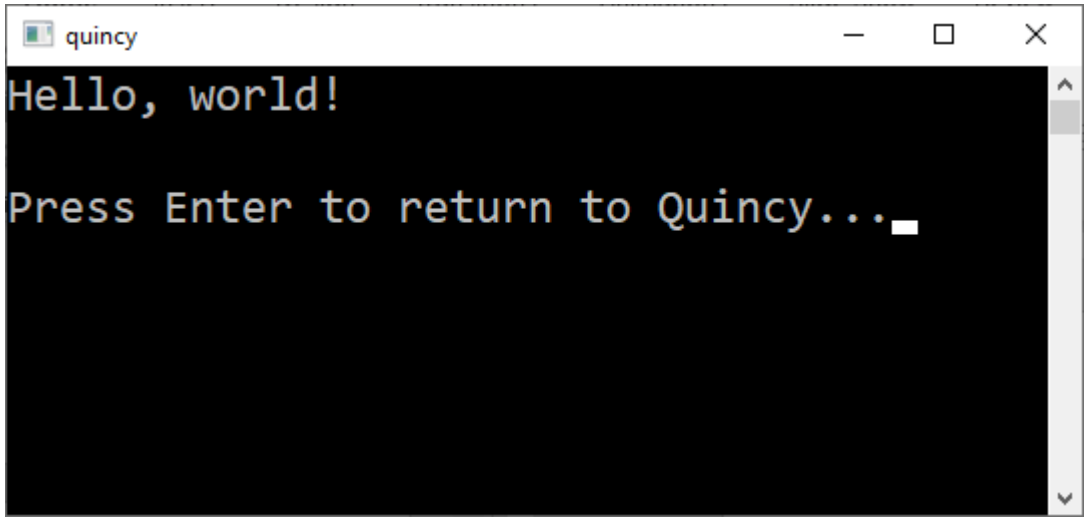
```
#include <stdio.h>
#define TEXT "Hello, world!\n"

int main(void)
{
   printf(TEXT);
   return (0);
}
```



Preprocessor Directives: #define

It's bad practice but... We can be silly:

```
#include <stdio.h>
#define HELLO main
#define SAY printf

int HELLO(void)
{
   SAY("Hello, world!\n");
   return (0);
}
```



- This makes perfect sense when you understand how **#define** works.
- Don't do things like this, though.
- It just obfuscates your code.

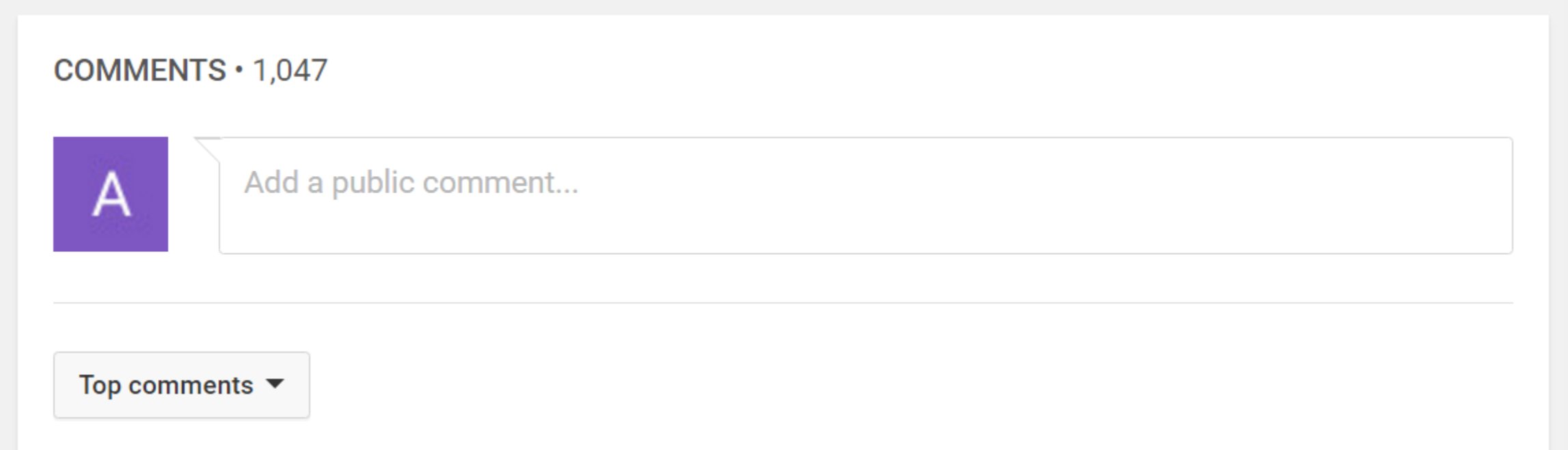
Preprocessor Directives: `#define`

A far better use for **`#define`**:

```
#include <stdio.h>
#define PI 3.14159265359

int main(void)
{
   printf("Hello, world!\n");
   return (0);
}
```

- This a good use of **`#define`**.
- Numerical or physical constants that never change.
- Once something is **`#defined`**, it cannot be changed later.
- **`#defined`** values are fixed at compile time.
- Convention is to use CAPS

Comments



Comments

```
/* This is a comment */

/* So is this.
   It can be spread
   over multiple lines */
```

Comments begin with `/*` and end with `*/`

Comments

```c
#include <stdio.h>
/* define PI */
#define PI 3.14159265359

int main(void) /* main function */
{
   /* Say hello */
   printf("Hello, world!\n");
   return (0);
}
```

- Comments can be used to explain your code to other programmers
- They can also serve as reminders for the original programmer.
- They are NOT compiled.

Pop Quiz

Is this comment formatted correctly?

/* This is a very long comment.
It takes up /* multiple lines */

Yes.

This /* is part of the comment, and therefore not compiled.
Adding a redundant /* does not negate the first /*.

Pop Quiz

Is this comment formatted correctly?

/* This is a very long comment.
It takes up */ multiple lines */

No.

This `*/` closes the comment. Thus, "`multiple lines */`" is no longer inside the comment and gets compiled (error).

Statements & Semicolons

In C, statements are terminated with a semicolon:

```
#include <stdio.h>   ?

int main(void)
{
   printf("Hello, world!\n");   /* printf statement */
   return (0);   /* return statement */
}
```

Preprocessor directives are not statements! Don't need semicolon.

Line Breaks? Spaces? Tabs?

They don't matter in C. However….

```
#include <stdio.h>

int main(void)
{
   printf("Hello, world!\n");
   return (0);
}
```

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

One of these is much easier to read.
Readability is very important when writing code.

Line Breaks? Spaces? Tabs?

Quincy 2005 - [HelloWorld.c]

File Edit View Project Debug Tools Window Help

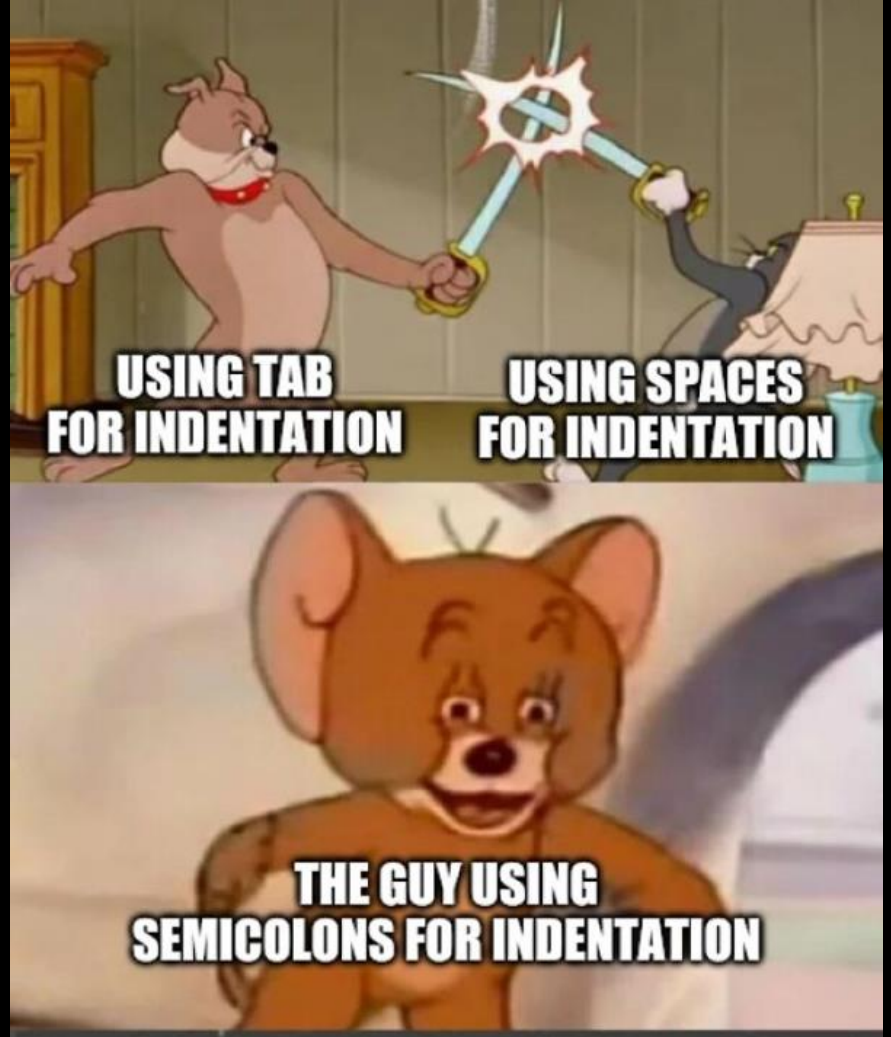
```
#include <stdio.h>
    int main(void)        {
printf("Hello, world!\n");
return (0)          ;}
```

Press F1 for help

quincy

```
Hello, world!

Press Enter to return to Quincy...
```

Bad practice.




```
int main (int argc, char* arg[]) {
;;;;int i = 0
;;;;while (i < 10) {
;;;;;;;;printf("%d\n", i)
;;;;;;;;i ++
;;;;}
;;;;return 0
;}
```

The International Obfuscated C C

ioccc.org

Logo by winner Matt Zucker

The International Obfuscated C Code Contest

[The judges | **IOCCC home page** | How to enter | FAQ | Mirrors | IOCCC news | People who have won | Winning entries]

The winners of the 27th IOCCC have been announced. Congratulations!

Please see the following news items.

Goals of the Contest

Obfuscate: tr.v. -cated, -cating, -cates.

1. a. To render obscure.
   b. To darken.
2. To confuse: his emotions obfuscated his judgment.

[LLat. obfuscare, to darken : ob(intensive) + Lat. fuscare, to darken < fuscus, dark.] -obfuscation n. obfuscatory adj

The IOCCC:

- To write the most Obscure/Obfuscated C program within the rules.
- To show the importance of programming style, in an ironic way.
- To stress C compilers with unusual code.
- To illustrate some of the subtleties of the C language.
- To provide a safe forum for poor C code. :-)

IOCCC news

2020-07-25

- The winners of the 27th IOCCC have been announced. Congratulations!
- The winning code is being reviewed and will be published in mid-August

2020-03-22

- Due to current world events, we are extending the 27th IOCCC until 2020-May-15 06:26:49 UTC
- The submission tool will now be active until 2020-May-15 06:26:49 UTC

https://www.ioccc.org/2019/dubl

ioccc.org/2019/duble/prog.c

```
#include <sys/time.h>
#include <termios.h>
#include <unistd.h>
#include <fcntl.h>

int a,b,c,d,e,f,h,i,j,k,l,m,n;fd_set p[1],q;struct{int A,B,C,D,E,   F
,G,H,I,J,L,K;}w  ,x;r(int r){return b=r,FD_ISSET(r,p);}enum{u=V*( 11 *W
+5)+16};t( int    t){FD_SET(t,&q);}struct sockaddr_un A,  B,*D,C={       .
sun_family=1};    char v[u];unsigned int E=sizeof(B),F=   sizeof(          x
);char*I="\e"     "[0m\e[2J\e[0;0H",*K,*L=&C.sun_path,    *M,*J           =
"\e[0;30;40"      "m \0\e[0m\n\0#!/bin/cat\n\e[1A",N;     O(){          n=
0;while(++n<      9)*(n+L)=rand()%13+65;t(b=socket(1      ,2           ,0
));n=bind(b,      b==4?&B:&C,E);}struct timeval y,z      ;H           (
){recvfrom(b      ,&w,F,0,&A,&E),a=w.A;}Q(){close       (           4)
;fopen(K,"r" )    &&(read(4,v,u),close(4),unlink      (K           )
),O();b=3;}P (    int P){strcat(v,J+P);}G(int G      ){          x.
A=G;D=G<5?0: G    ^8?&B:&A;return sendto(b,G^3      ?           &
x:&w,F,0,D,! !    D*E);}struct termios o;main      (          int
F,char**D){n =    0;for(P(18);n++<V;P(12))//     !           b+12
for(b=0;b++< W    ;)P(0);if(F==2)for(K=D[1     ],           t(0),
srand(getpid (    )),strcpy(L,K),B=C,*L=s     ,           x.J=44,
O(),Q(),G(6);F    ++-4;)for(tcsetattr(1,    0,           (//IOCCC
tcgetattr(1,&o    ),o.c_lflag^=g,&o));F    -4          ;) {//2019
gettimeofday(&    y,0);n=y.tv_usec;m||    (           m   =(3<<19
)+n);z.tv_usec    =(m-n)%999999;b=3;    *p          =q    ;select
(5,p,0,0,&z)||    (x.E^=1,G(9),m=0)    ;           r(     0)&&(b=
3,read(0,&N,1)    ,N%=65,34^N||(x.   D=           x.      D%6+1),
N^47&&N^43||(x    .J=N==x.J?x.E=0   ,           G(10      ),44:N)
,N<4&&(x.E=0,G    (9),x.F+=N==2&  (x           .F-=N      ==3&&x.
F)!=W-3,x.G+=N    ==1&(x.G-=! N  &&          x.G)!=V      -3,x.E=
1),x.J^43&&(x.    H=x.F,x.I=x   .           G),G(9),      48^N||(
x.E=0,G(9),G(7    )));if(r(4  ))          if(H(),a^6      )if(a^5
)while(++b<=c)    G(3);else  x           .K=v[w.L],b      =w.B,G(
4);else/*O()*/    connect( (O           (),x.C=b),&A      ,E),b>c
&&(c=b),G(1) ;    if(r(3) )           if(H(),a^2){ a      ||G(6),
a^1||(x.B=w.C,    x.D||(x.           D=x.B%6+1),x. L      ||G(5))
,a^4||(v[x.L++]  =w.K, x.          L<u?G(5):printf (      "%s%.*"
"s\r \e[97mmod"  "e:" ,           I,u-6,v));if(a==7)      if(x.B^
5||close(w.B),w.B==x.B          ){if(x.B==5)for(/**/      unlink(
K),write(creat(K,511           ),v,u),b=5;++b<=c;)if      (G(2)+1
)for(H();++b<=c;)G(           0);printf(I);break;}if      (a>8)//
for(d=w.F,e=w.G,f           =w.H,h=w.I,a=w.E,i=d<f?1      :-1,j=(
f-d)*i,k=e<h?1:-           1,l=(h-e)*k,n=(j>l?j:-l)/      2;M=34+
(v+(e+1)*(5+W *          11                              )+d*11)
,47^w.J&&43^w           .                                J|w.A ^
10||(*M=w.D+          48                                 ),*(M-3
)=w.D*(w.D           +48!=*M)+48,*(M-6)=w.E*7+48,*(M+2)=a?42:32 ,
printf(""           "\e[%d;%dH%.11s\e[%d;%dH\e[0;4%dm%c%s%s",e+2,
d+2,M-8           ,V,8,x.D,x.J+65,I+8,J),f^d|e^h;)(b=n)>-j&&(n-=l
,d+=i)           ,b<l&&(n+=j,e+=k),a=0;fflush(stdout);}else//d+=i
   0+          Q(
  )           ,
 G          (6
   ),      G(
      8)  ;
        }}
```

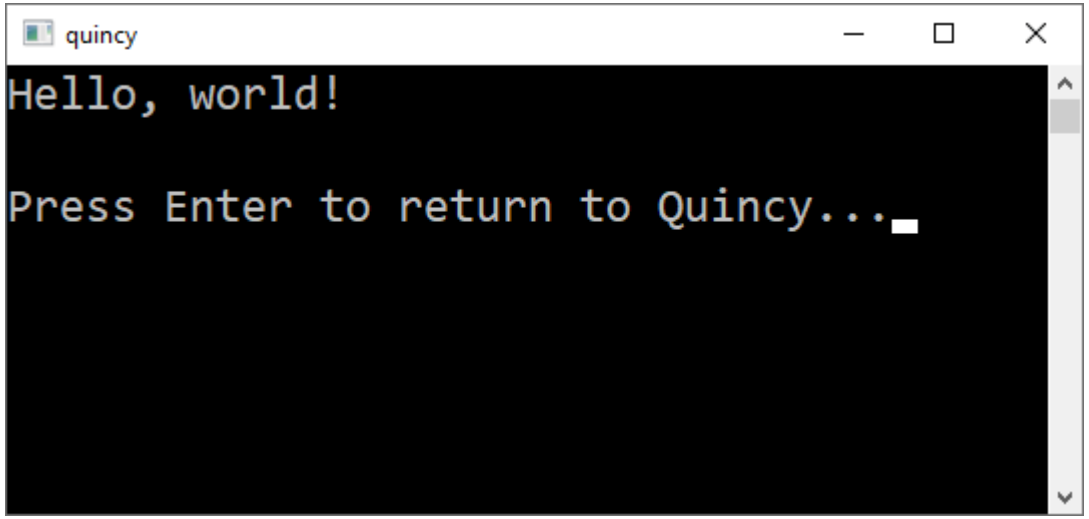
Skeleton of a Program

```
#include <stdio.h>
/* Optional additional includes go here */
/* Optional #define macros go here */

int main(void) /* Every C program starts with main() */
{
   /* Optional declarative (?) statements go here */
   /* One or more executable statements start here */
   printf("Hello, world!\n");
   return (0);
}
```

Skeleton of a Program

One or more executable statements?

```
#include <stdio.h>

int main(void)
{
    printf("Hello");
    printf(", ");
    printf("world");
    printf("!\n");
    return (0);
}
```



Skeleton of a Program

Declarative statements? Refers to ***variable declarations***

- A variable is a name that refers to a memory location.
- Variables store data – Think of them as containers for values.
- Variables must be declared prior to usage.
- Three basic types: `int`, `double` (or `float`), `char`
- **`double`** is double-precision floating-point; **`float`** is single-precision floating point.

Variable Declaration

```
#include <stdio.h>

int main(void)
{
    int wholeNum;
    double realNum;
    char charValue;
    return (0);
}
```

?

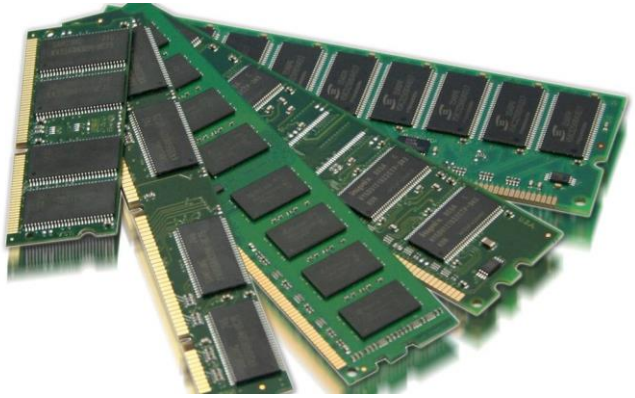
```
// Reserve memory for int
// Reserve memory for double
// Reserve memory for char
```

Memory?

2052	10100011
2053	00001000
2054	11001100
2055	10001100
2056	10001101
2057	11001101
2058	00101010
2059	10001100

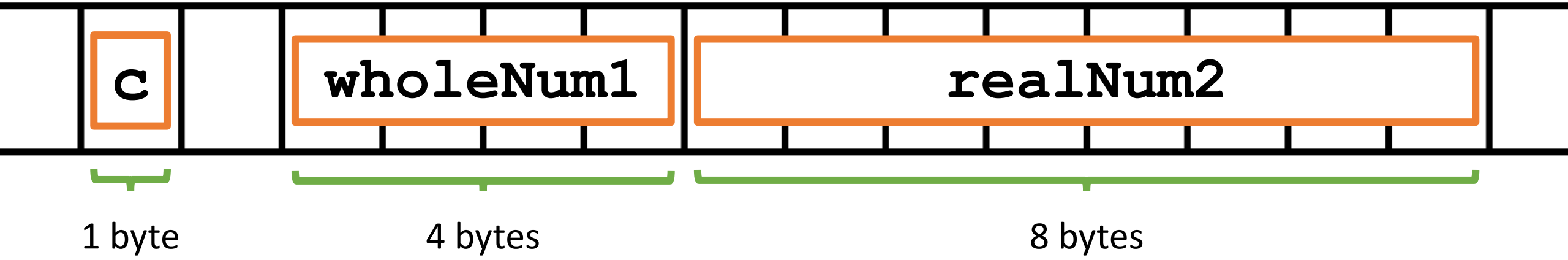
Computer memory:

- A collection of consecutively numbered cells
- Each cell is typically one byte
- One byte == 8 bits



Variables in Memory

```
char c;                /* 1 byte allocated  */
int wholeNum1;         /* 4 bytes allocated */
double realNum2;       /* 8 bytes allocated */
```



More on Memory

- We don’t control where our variables go in memory.
- This is handled by the Operating System
- We can use *pointers* to manipulate our program’s memory directly – we’ll see these later in the course.
- **Remember:** Think of variables as containers in memory.
- A variable’s name (identifier) is how we access and modify its location in memory.

Variable Declaration

```
#include <stdio.h>

int main(void)
{
    int wholeNum;
    double realNum;
    char charValue;
    return (0);
}
```

Format:
`<type> <identifier>;`

Type:
`int`, `float`, `double`, `char`, etc.

Identifier (variable name):
`wholeNum`, `realNum`, `charValue`

Remember to end statements with semicolons!

Declaration & Initialization

```
#include <stdio.h>

int main(void)
{
    int num1 = 15, num2;
    double realNum = 3.14159;
    char charValue = ‘$’;
    charValue = ‘a’;
    num2 = 42;
    return (0);
}
```

- We can initialize variables in the declaration.
- We can mutate them later using their identifiers.
- Variables must be declared before we can use them.

Variable Types in C

Integer types

Floating-point types

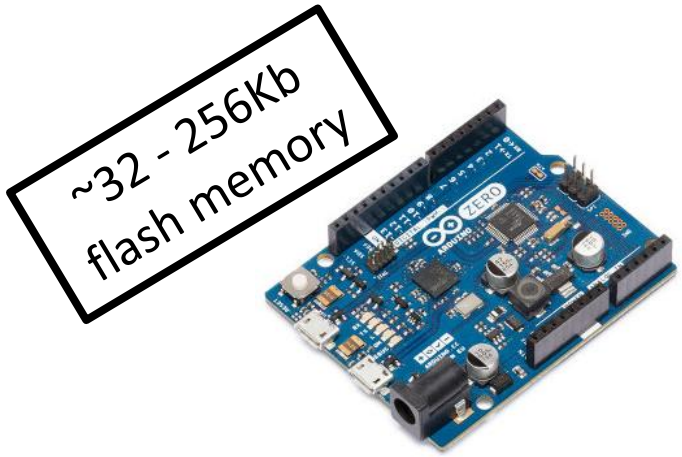
Name	Description	Size*	Range*
`char`	Character	1 byte	signed: -128 to 127 unsigned: 0 to 255
`short int` (`short`)	Short Integer.	2 bytes	signed: -32768 to 32767 unsigned: 0 to 65535
`int`	Integer.	4 bytes	signed: -2147483648 to 2147483647 unsigned: 0 to 4294967295
`long long int`	Long integer.	8 bytes	signed: -2^63+1 to +2^63-1 unsigned: 2^64 - 1
`float`	Floating point number.	4 bytes	+/- 3.4e +/- 38 (~7 digits)
`double`	Double precision floating point number.	8 bytes	+/- 1.7e +/- 308 (~15 digits)

(*depends on OS)

What's the Point?

`short int` (`short`)	Short Integer.	2 bytes	signed: -32768 to 32767 unsigned: 0 to 65535
`int`	Integer.	4 bytes	signed: -2147483648 to 2147483647 unsigned: 0 to 4294967295
`long long int`	Long integer.	8 bytes	signed: -2^63+1 to +2^63-1 unsigned: 2^64 - 1

What's the Point?



The obvious:

- Space is very limited on embedded systems
- You should represent data as efficiently as possible

The not-so-obvious:

- Even 128GB of RAM fills fast when you're editing HD video
- Multi-TB hard drives fill fast when you're storing large data sets.

Overflow

YouTube

Shared publicly - Dec 1, 2014

We never thought a video would be watched in numbers greater than a 32-bit integer (=2,147,483,647 views), but that was before we met PSY. "Gangnam Style" has been viewed so many times we had to upgrade to a 64-bit integer (9,223,372,036,854,775,808)!

Identifiers

The term “identifier” refers to the name of a variable (or function – i.e. the **`main`** function).

There are rules for what you can name your variables:

Must contain only letters, digits, or underscores
Must never begin with a digit
Must not be a reserved word

Identifiers are case sensitive!

Must contain only letters, digits, or underscores
Must never begin with a digit

Legal:	**Illegal:**
_ABC123	-ABC123
A_b_c_123	123abc
hello_world	67_q
HELLO_WORLD	H@T
X1	X+1
Y78	Y^78
thisIsMy1stVariable	Thi%IsMy1$tV@riable

Must not be a reserved word


```
auto        else        long        switch
break       enum        register    typedef
case        extern      return      union
char        float       short       unsigned
const       for         signed      void
continue    goto        sizeof      volatile
default     if          static      while
do          int         struct      _Packed
double
```

Remember case sensitivity! The following are all legal, though bad practice:

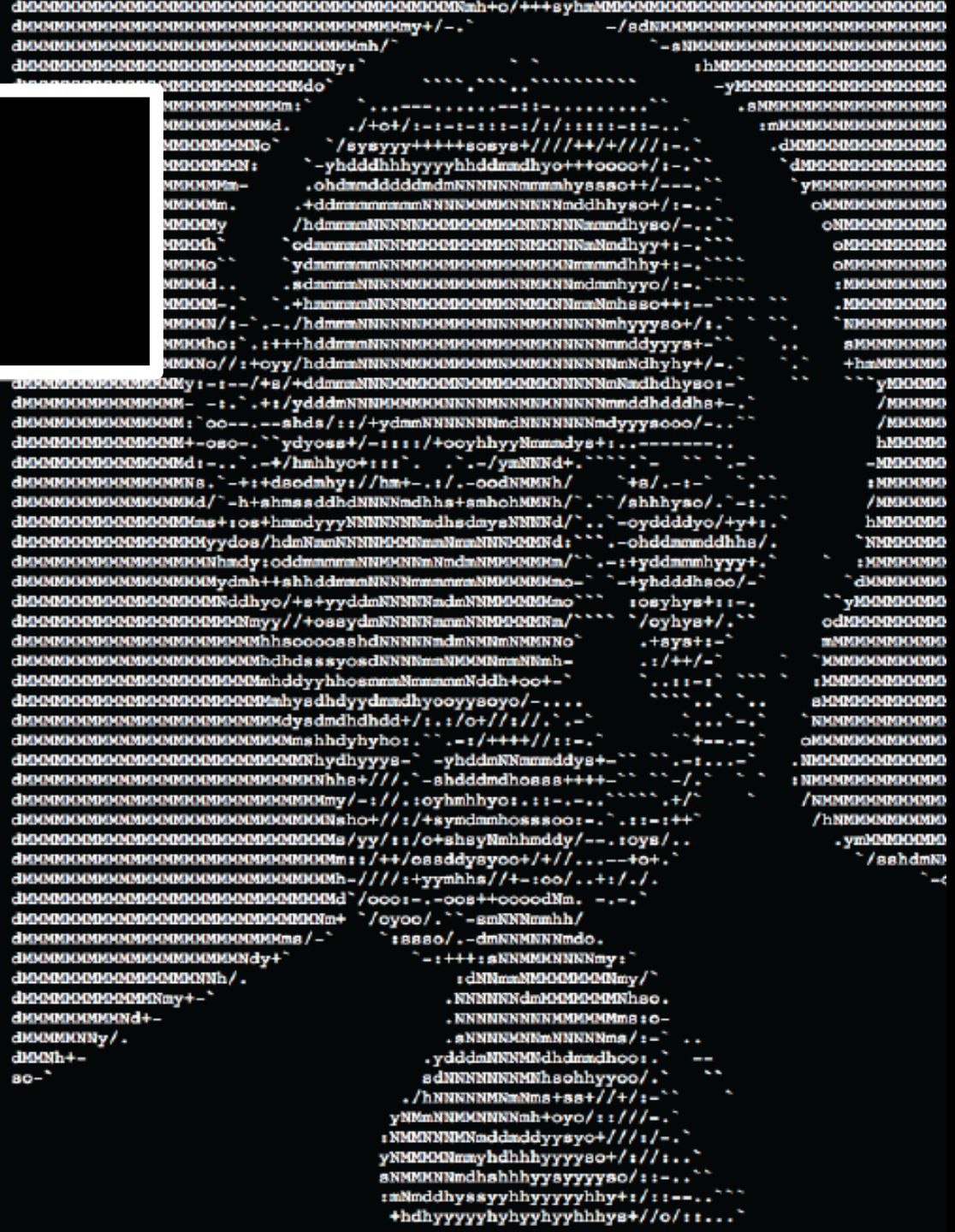
INT, Char, DOUble, eXtern, VoID, strucT, wHiLe, and so on.

Pop Quiz

Which are valid identifiers?

`Abc`	`_32`	~~`32x`~~	~~`Price$`~~	~~`file?1`~~
~~`int`~~	`abc87`	~~`45%`~~	~~`return`~~	`CHAR`

Formatted Output



printf

print-formatted

```
int number = 15;
printf("Number is %d \n", number);
```

Everything inside the quotes gets printed to the screen, though not exactly as it appears.

%d is a ***placeholder***. It means, here we want to print the value of a variable. Specifically, **%d** is for integer.

Beyond the quotes, after a comma, is the name of the variable we want to print.

Formatting Placeholders

```
int temp = 9;
printf("It is %5d degrees\n", temp);
```

%5d Reserves 5 places (**right-justified**)

```
Output
   It is ____9 degrees
```

4 spaces (not underscores), one spot for the number

Formatting Placeholders

Literal!

```
printf("Number is %-8d.\n", 1234);
```

%-8d reserves 8 places (**left-justified**)

```
Output
    Number is 1 2 3 4 _ _ _ _.
```

4 spots for the number, 4 spaces after

Formatting Placeholders

```
int num = -38;
printf("%6d\n", num);
```

```
Output
   ___-38
```

The **negative sign** takes up a position!

More Placeholders

Seen:	**%d**	decimal (`int`)
Also:	**%f**	floating-point (`float`)
	%lf	long floating-point (`double`)
	%c	character (`char`)
	%s	string (later in the course)

<u>Common pitfall:</u>

- **d** is for **d**ecimal, not double.
- Use **%d** for int, **%lf** for double.

Formatting Floating-Point (double/float)

```
double num = 1.21997;
printf("%lf", num);
```

`%lf` placeholder for `double`

```
Output
    1.219970
```

```c
#include <stdio.h>

int main(void)
{
    double num = 3.1415;

    /*1*/ printf("%lf\n", num);
    /*2*/ printf("%.2lf\n", num);
    /*3*/ printf("%8.2lf\n", num);
    /*4*/ printf("%.6lf\n", num);
    /*5*/ printf("%-8.2lf\n", num);
    /*6*/ printf("%-.3lf\n", num);
    return (0);
}
```

1) %lf

Default format (6 decimal places)

2) %.2lf

Two decimal places (it rounds!)

3) %8.2lf

Two decimal places, right justify using 8 spaces total.

4) %.6lf

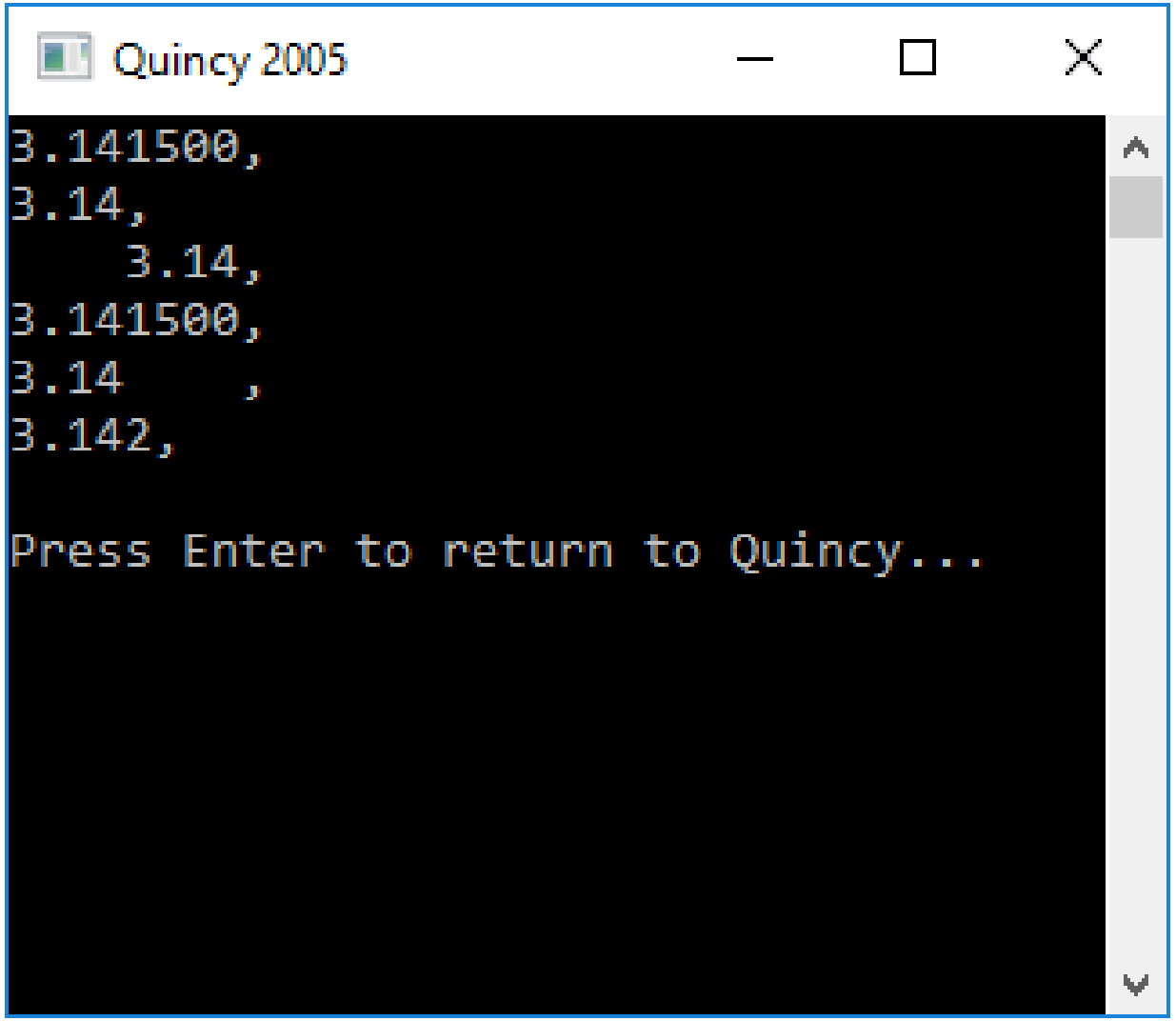
Six decimal places, right justify

5) %-8.2lf

Two decimal places, left justify using 8 spaces total.

6) %-.3lf

Three decimal places, left justify



1) %lf

Default format (6 decimal places)

2) %.2lf

Two decimal places (it rounds!)

3) %8.2lf

Two decimal places, right justify using 8 spaces total.

4) %.6lf

Six decimal places, right justify

5) %-8.2lf

Two decimal places, left justify using 8 spaces total.

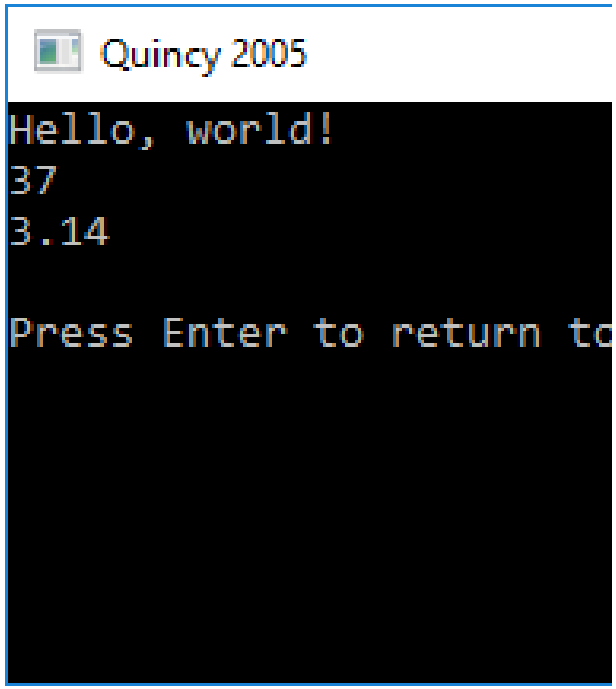
6) %-.3lf

Three decimal places, left justify

Many Variables, One printf

```
#include <stdio.h>

int main(void)
{
    double num1 = 3.1415;
    int num2 = 37;
    char c1 = ‘H’;
    char c2 = ‘w’;
    char c3 = ‘!’;

    printf(“%cello, %corld%c\n”, c1, c2, c3);
    printf(“%d\n%.2lf\n”, num2, num1);
    return (0);
}
```

Placeholders



Mismatched Placeholders

```
int number = -27;
printf("Your number is: %f", number);
```

- Here, we are printing an integer variable using a float placeholder.
- In memory, the binary bit pattern represents an integer.
- In our code, we're telling printf to read it as a float (%f)
- Integer and float are represented very differently in binary!

This is not a syntax error!

- This code will run, but it will print the wrong value.

Mismatched Placeholders

Quincy 2005 - [HelloWorld.c]

File Edit View Project Debug Tools Window Help

```c
#include <stdio.h>

int main(void)
{
    int num = -27;
    printf("num: %f\n", num);
}
```

Press F1 for help Ln 9, Col 1

quincy

```
num: -1.#QNAN0

Press Enter to return to Quincy...
```

A Brief Aside: Escape Sequences

<u>A small sampling:</u>

```
\n   Console newline
\t   Console tab
\b   Console backspace
\a   Produces beep/tone
\”   Prints ”
\’   Prints ’
\\   Prints \
```

Questions?