“We found that students’ mindsets—how they perceive their abilities—played a key role in their motivation and achievement, and we found that if we changed students’ mindsets, we could boost their achievement. More precisely, students who believed their intelligence could be developed (a growth mindset) outperformed those who believed their intelligence was fixed (a fixed mindset). And when students learned through a structured program that they could “grow their brains” and increase their intellectual abilities, they did better. Finally, we found that having children focus on the process that leads to learning (like hard work or trying new strategies) could foster a growth mindset and its benefits.”

-- Carol Dweck, author of Mindset: The New Psychology of Success

https://www.edweek.org/leadership/opinion-carol-dweck-revisits-the-growth-mindset/2015/09

Basic C

CSCI 237: Computer Organization
2nd Lecture, Monday, Sept. 9

Kelly Shaw

Administrative Details

- Make sure to introduce yourself on Slack if you haven't already!
- Lab #1 checkpoint (Parts 0 and 1) due Wednesday at 11pm
  - Any questions?
- Read CSAPP 2.1 for Wednesday
- My help hours today
  - 1-2:30pm in TCL 309 (if not there, find me in TCL 312 "majors lab")

Last Class

- Course logistics
- Hello World

Today's Plan

- printf
- Compilation of C programs
- C syntax and examples
  - Primitive types
  - Operators
  - Arrays
    - Command line arguments
  - Reading inputs from stdin (i.e., scanf)

C Functions

- Global functions
  - ex. *main*
- Return type specified before function name
- Parameters specified w/ type in parentheses

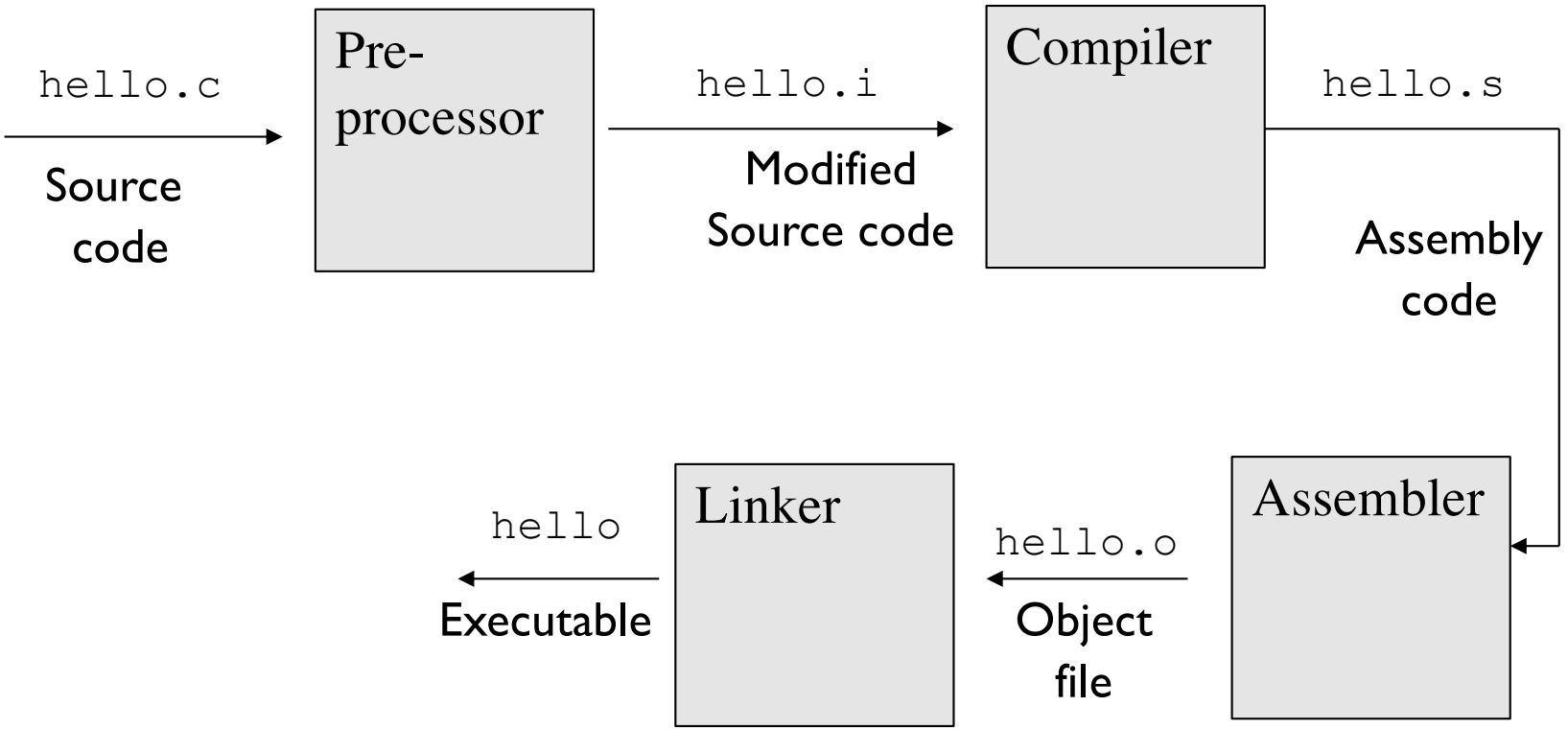
```
#include <stdio.h>

int foo(int x) //definition
{
 return 2*x;
}

int main(int argc, char *argv[])
{
 printf("Foo %d\n", foo(3)); //call
 return 0;
}
```

printf

```
printf(<formatted string>, …)
```

- Prints to `stdout`
- Specify formatted string as 1st argument. Use `%` to indicate type for each value to be inserted.
- Remaining arguments are items to be placed in string
  - `printf("hello world!");`
  - `printf("int num %d float num %f %s\n", 3, 3.14, "done");`
- Part of the `stdio.h` library
  - Insert `#include <stdio.h>` at top of C file

Program Compilation




```
gcc -o hello hello.c
```

Variables and Data Types

- C requires you to specify the type for every variable
- Primitives:
  - `bool` `// #include <stdbool.h>`
  - `char`
  - `short`
  - `int`
  - `long`
  - `float`
  - `double`
- `sizeof(…)`
- Integral types also have `unsigned` versions
  - Must take care with comparisons between signed and unsigned!!!

Type specifies # of bits/range of values and interpretation of bits

Literals

- bool
  - true / false
- int
  - 38
- long
  - 38L
- float
  - 3.14f
- double
  - 3.14
- char
  - 'a'
- C-style string
  - "Hello"

Arithmetic Operators

- Addition
  - `val = 3 + 2;`
- Subtraction
  - `val = 3 - 2;`
- Multiplication
  - `val = 3 * 2;`
- Division
  - `int val = 3 / 2;`
  - `double dval = 3.0/2.0;`
- Modulo / remainder
  - `val = 4 % 3;`

- Operator with assignment
  - `+=, -=, *=, /=`
  - `val += 3;`
- Auto-increment/decrement
  - `x++;`
  - `++x;`
  - `x--;`
  - `--x;`

Including auto-inc/dec in larger expression is poor style
Poor style example: int y = x++ / 3;

HelloWorld' – Print (1+2)

```
#include <stdio.h>

int main(int argc, char *argv[])
{
    printf("%d\n", (1+2));
    return 0;
}
```

HelloWorld'' – Print x + y

```
#include <stdio.h>

int main(int argc, char *argv[])
{
    int x = 1, y = 2;

    printf("%d\n", (x+y));
    return 0;
}
```

Arrays: Collections of Data

- In addition to individual variables, collections of data can be allocated **statically**
- Indices start at 0
- There is no length field
  - You have to remember the size of the array
- Size cannot be changed

```
int main(int argc, char *argv[])
{
  int intArray[3];

  intArray[0] = 0;
  intArray[1] = 1;
  intArray[2] = intArray[1]+1;

  return 0;
}
```

Command Line Arguments

```
int main(int argc, char *argv[])
{
        …
}
```

- `argc` : number of C-style strings in the array
- `argv` : array of C-style strings
- First argument (index 0) is always the program name

Command Line Argument: argv[0]

```
#include <stdio.h>

int main(int argc, char *argv[])
{
    printf("%s\n", argv[0]);
    return 0;
}
```

ASCII to int: atoi(const char *str)

```
#include <stdio.h>
#include <stdlib.h> // contains atoi(…)

int main(int argc, char *argv[])
{
    char *numStr = "142";
    // char numStr[4] = "142";
    int num = atoi(numStr);

    printf("%s %d\n", numStr, num);

    return 0;
}
```

```
int atoi(const char *str) in stdlib.h
```