CSE 220 – C Programming

Writing Large Programs

```c
#include <stdio.h>
void say_hi(void) {
   printf("Hi!\n");
}

int main(void) {
   say_hi();
   return 0;
}
```

What is the output?

1. Hi!
2. bye
3. I'm just a silly person
4. Error

```c
#include <stdio.h>

int main(void) {
   say_hi();
   return 0;
}

void say_hi(void) {
   printf("Hi!\n");
}
```

What is the output?

1. Hi!
2. bye
3. I'm just a silly person
4. Error

```c
#include <stdio.h>
void say_hi(void);
int main(void) {
   say_hi();
   return 0;
}

void say_hi(void) {
   printf("Hi!\n");
}
```

What is the output?

1. Hi!
2. bye
3. I'm just a silly person
4. Error

```c
#include <stdio.h>
void say_hi(void);
int main(void) {
    say_hi();
    return 0;
}

void say_hi(void) {
    printf("Hi!\n");
}
void say_hi(void) {
    printf("Hi!\n");
}
```

What is the output?

1. Hi!
2. Hi! Hi!
3. ???
4. Error

Writing Large Programs

- Typical for programs to consist of multiple files
  - Source files
  - Header files
  - Building (compiling and linking) a program

Source Files

- A program may be divided among several source files
  - Grouping related function in one file clarifies the program structure
  - Each source file can be compiled separately
  - Easier to reuse code
- By convention, the extension is ***.c***
- One source file must contain the function main, the starting point of the program

Example

ArrayDisplay.c

```
void printSingle(int array[], int n) { ... }
void printMultiple(int array[], int n) { ... }
void printSeparated(int array[], int n, char sep) { ... }
void printAbove(int array[], int n , int val) { ... }
void printBelow(int array[], int n , int val) { ... }
```

ArrayControl.c

```
void insert(int array[], int n, int idx, int value) { … }
void delete(int array[], int n, int value) { … }
void deleteAll(int array[], int n, int value) { … }
int findPos(int array[], int n, int value) { … }
```

Example

MainProg.c

```
…
int main(int argc, char *argv[]) {
      int grades[100];
      insert(grades, 100, 0, 90);
      insert(grades, 100, 0, 70);
      print(grades, 100);
      return 0;
}
```

Header Files

- How can a function in one file call a function in a different file? Or access an external variable?
- `#include` directive allows sharing among files
- Files included using the #include directive are called header files.
- By convention, extension is ***.h***

```
#include <stdio.h>

#include <string.h>
```

Include directive

- Two forms:
  - `#include <filename>`
  - `#include “filename”`
- `#include <filename>`
  - Looks in directory where system headers reside
  - This is for the header files provided by the compiler
- `#include “filename”`
  - Search current directory, then searches directory containing system files
  - This is for the header files you write.

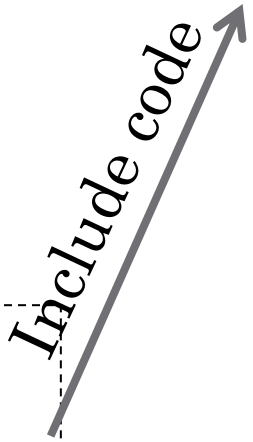
Code Sharing

- Share function declarations
- Share macro definitions (covered later)
- Share variable definitions (covered later)

Sharing function prototypes

ArrayDisplay.c

```
void printSingle(int array[], int
n) { ... }
void printMultiple(int array[],
int n) { … }
void printAbove(int array[], int n
, int val) { ...
void printBelow(int array[], int n
, int val) { ...
```



MainProg.c

```
…
…


int main(int argc, char
*argv[]) {
      int grades[100];
      insert(grades, 100,
0, 90);
      insert(grades, 100,
0, 70);
      print(grades, 100);
      printBelow(grades,
100, 60);
      return 0;
}
```

Sharing function prototypes

ArrayDisplay.h

```
void printSingle(int array[], int n);
void printMultiple(int array[], int n);
void printAbove(int array[], int n, int
val);
void printBelow(int array[], int n, int
val);
```

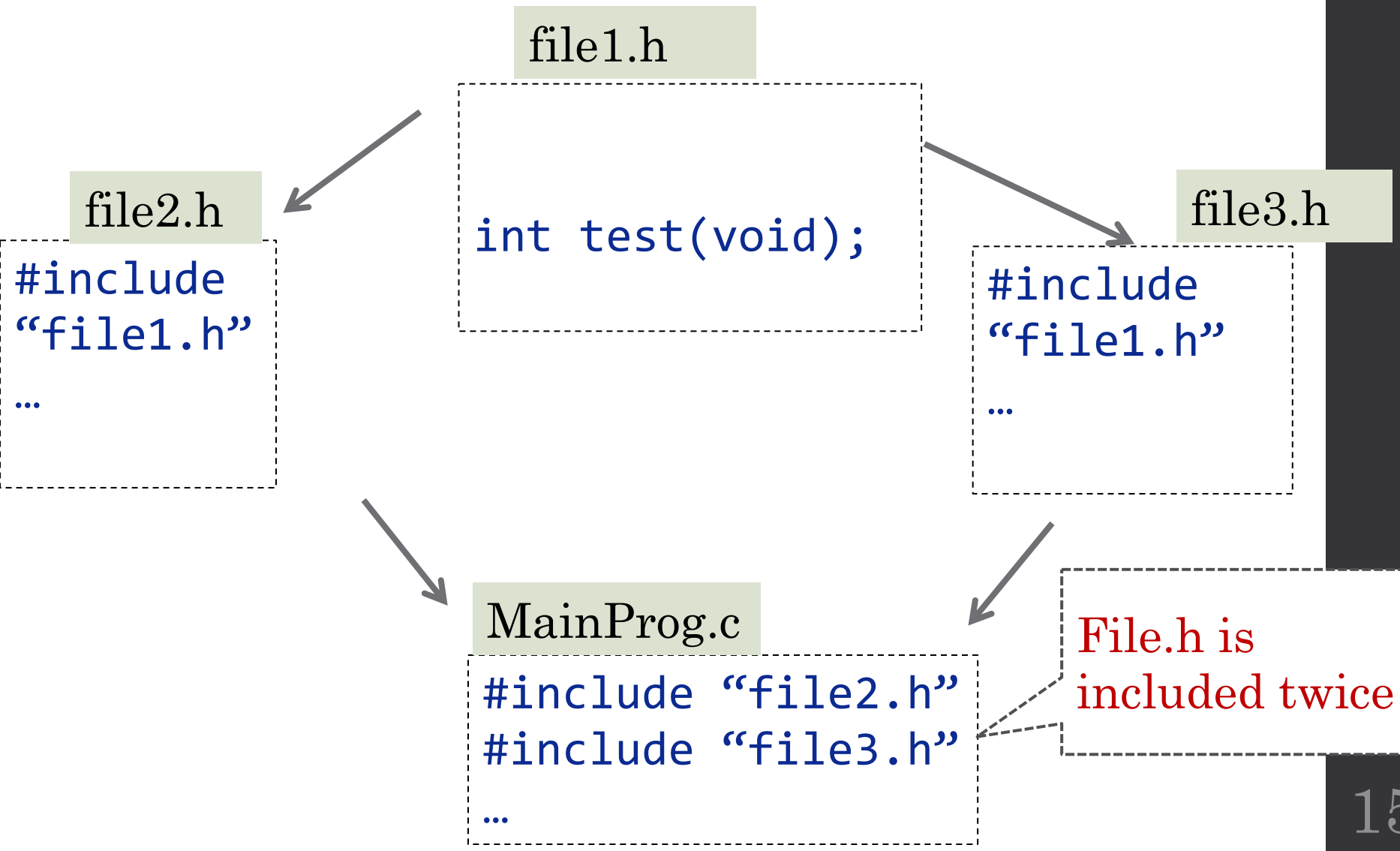
ArrayDisplay.c

```
#include “ArrayDisplay.h”

void printSingle(int array[], int n) { ... }
void printMultiple(int array[], int n) { … }
void printAbove(int array[], int n , int
val) { ... }
void printBelow(int array[], int n , int
val) { ... }
```

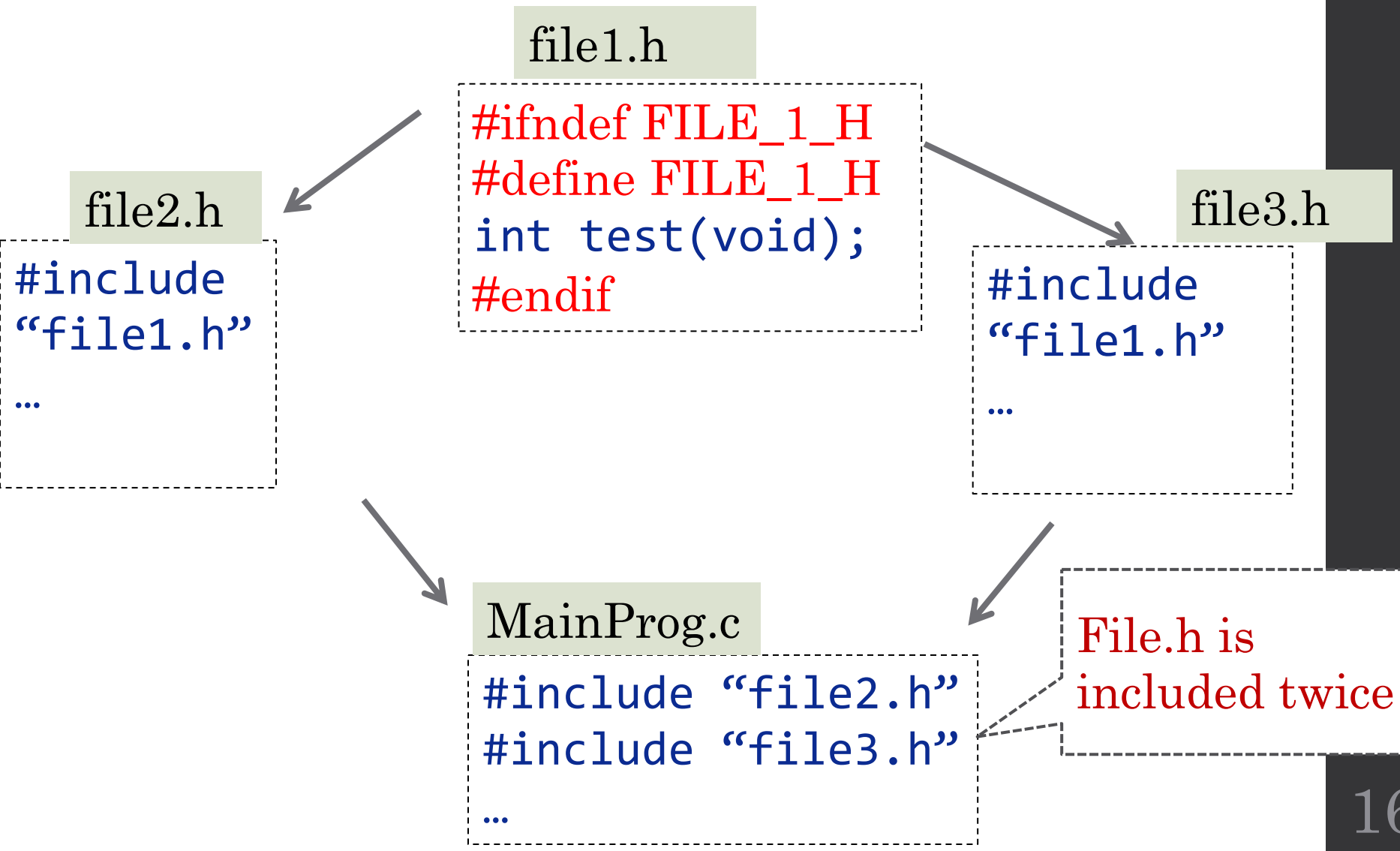
MainProg.c

```
#include
“ArrayDisplay.h”
…
…

int main(int argc, char
*argv[]) {
     int grades[100];
     insert(grades, 100,
0, 90);
     insert(grades, 100,
0, 70);
     print(grades, 100);
     printBelow(grades,
100, 60);
     return 0;
}
```

Protecting Header Files



Protecting Header Files



Protecting Header Files

file1.h

```
#ifndef FILE_1_H
#define FILE_1_H
…
int test(void);
#endif
```

MainProg.c

```
#include “file2.h”
#include “file3.h”
…
```

- The first time file1 is included, FILE_1_H is not defined.
- The second time file1.h is included:
  - FILE_1_H is defined
  - The preprocessor will not include the lines between #ifndef and #endif

Dividing into Files

ArrayDisplay.h

```
void printSingle(int array[], int n);
void printMultiple(int array[], int n);
void printSeparated(int array[],
   int n, char sep);
void printAbove(int array[], int n ,
int val);
void printBelow(int array[], int n ,
int val);
```

ArrayDisplay.c

```
#include "ArrayDisplay.h"
void printSingle(int array[], int n) { ...
}
void printMultiple(int array[], int n) { …
}
void printSeparated(int array[],
   int n, char sep) { … }
void printAbove(int array[], int n , int
val) {... }
void printBelow(int array[], int n , int
val) {... }
```

ArrayControl.h

```
void insert(int array[], int n, int
idx, int val);
void delete(int array[], int n, int
val);
void deleteAll(int array[], int n, int
val);
int findPos(int array[], int n, int
val);
```

ArrayControl.c

```
#include "ArrayControl.h"
void insert(int array[], int n, int idx,
int val) { ... }
void delete(int array[], int n, int val) {
... }
void deleteAll(int array[], int n, int
val) { ... }
int findPos(int array[], int n, int val) {
... }
```

Dividing into Files

ArrayDisplay.h

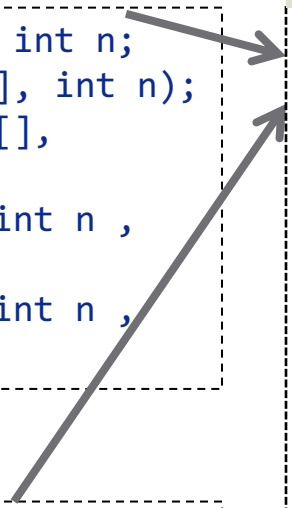
```
void printSingle(int array[], int n;
void printMultiple(int array[], int n);
void printSeparated(int array[],
   int n, char sep);
void printAbove(int array[], int n ,
int val);
void printBelow(int array[], int n ,
int val);
```

ArrayControl.h

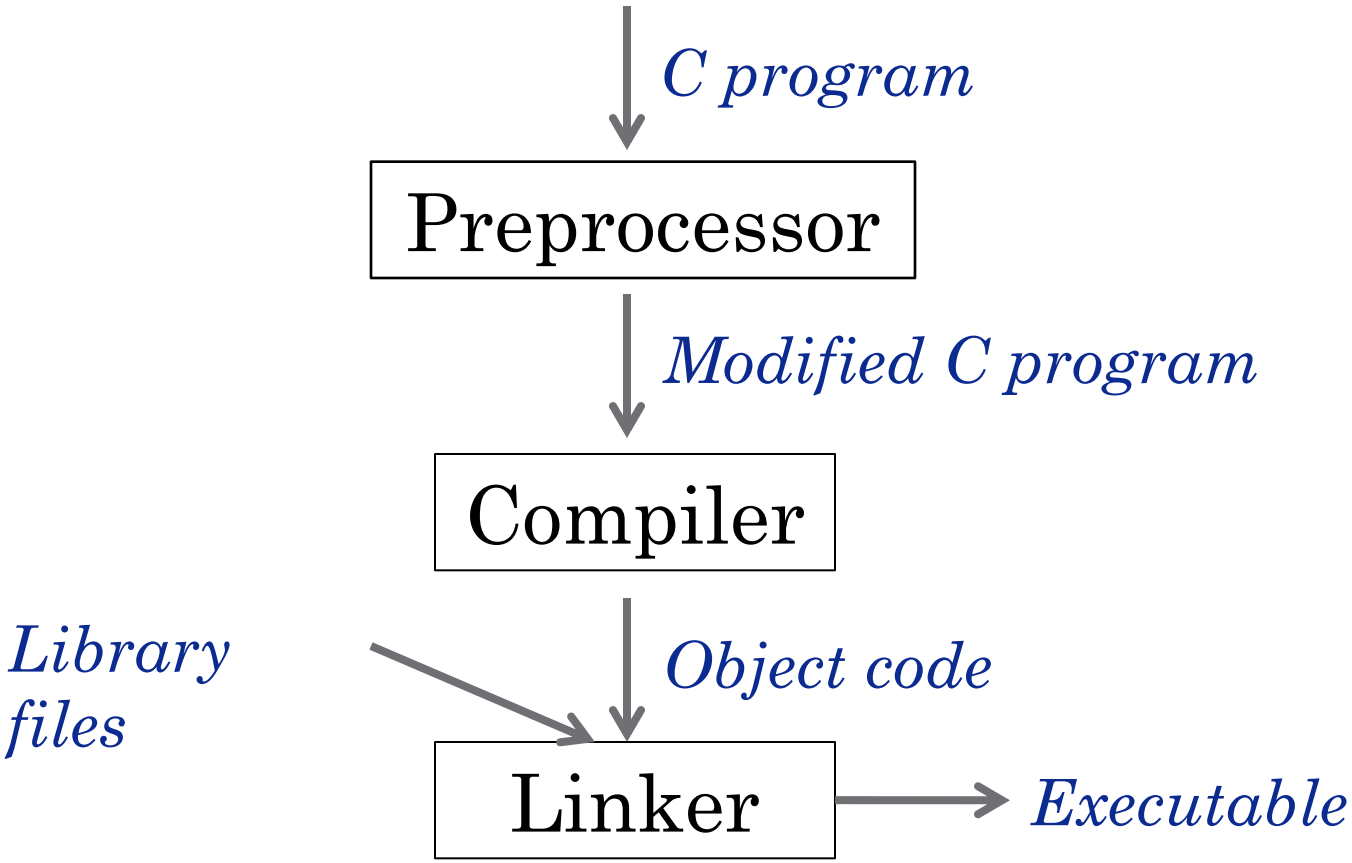
```
void insert(int array[], int n, int
idx, int val);
void delete(int array[], int n, int
val);
void deleteAll(int array[], int n, int
val);
int findPos(int array[], int n, int
val);
```

MainProg.c

```
#include “ArrayDisplay.h”
#include “ArrayControl.h”
….


int main() {
….


}
```



Building a Program



Building Multi-File Program

- Compilation:
  - Every source file must be compiled separately
  - Header files don't need to be compiled
  - An object file is generated for each source
- Linking:
  - Linker combines all object files and the needed library files and produces an executable
- Compilers allow building a program in one step:

```
gcc myprog.c arrayDis.c arrayCtrl.c –o myprog
```