NWEN 241
Systems Programming

Week 2 Tutorial

Functions

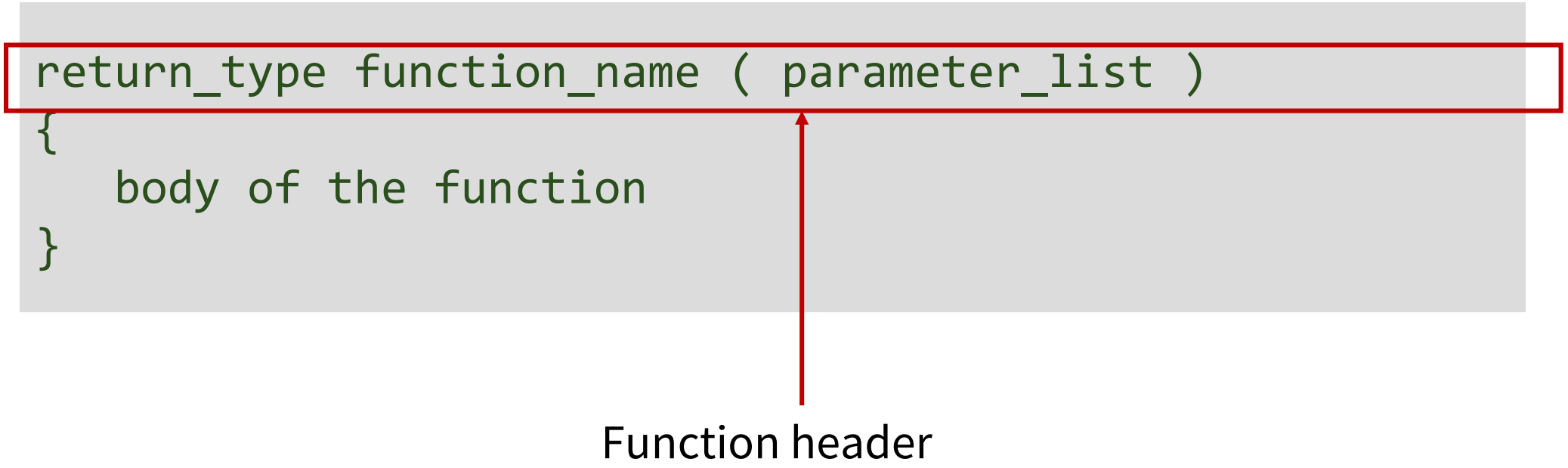
- Unlike Java, C allows functions to exist on their own, i.e., outside any class
  - In C, functions are first-class entities: a C program consists of one or more functions

- A C program must have exactly one `main` function
- Execution begins with the `main` function

Functions

- General form of a C **function definition:**

```
return_type function_name ( parameter_list )
{
   body of the function
}
```

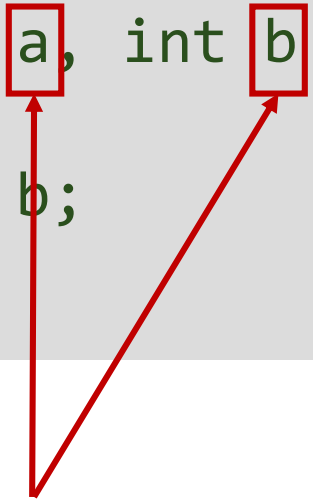


Functions

- Examples

```
void say_hello ( void )
{
   printf("Hello");
}
```

```
int add ( int a, int b )
{
   return a + b;
}
```



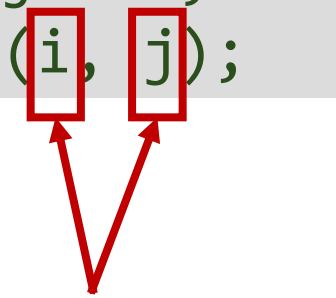
Formal parameters

Invoking Functions

- Example function invocations:

```
say_hello();
```

```
int i = 1, j = 2;
int k = add(i, j);
```



Actual parameters

- Before a function can be invoked, either the **function definition** or **function prototype** should have been declared prior to the invocation

Function Prototype

- A declaration specifying the return type, function name, and list of parameter types

```
return_type function_name ( parameter_types_list );
```

Function Prototype

- Examples

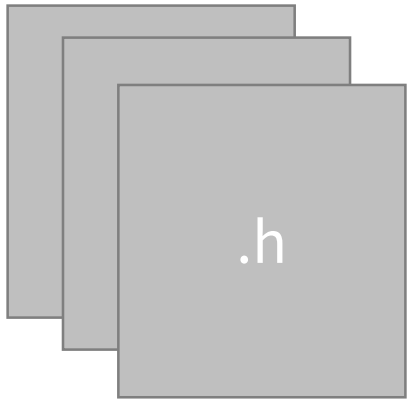
```
void say_hello ( void );
```

```
int add ( int a, int b );
```

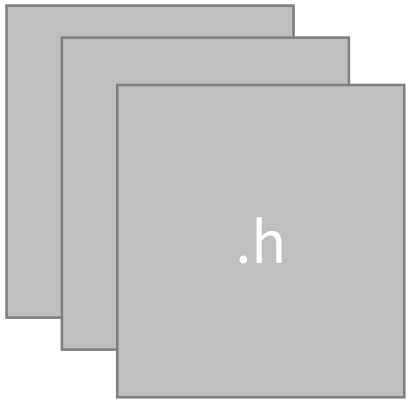
- No need to provide identifiers to input parameters, the types of the input parameters are sufficient

```
int add ( int, int );
```

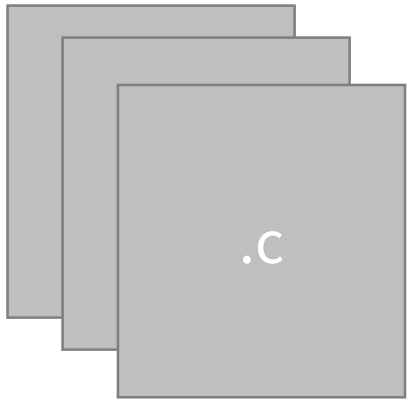
Recall: Large C Program



Header files
from standard
C library



Own header files



Source files

Function-Like Macro

Macro	Example Problematic Usage
`#define SQ(x) x * x`	`SQ(1+1)`
`#define SQ(x) (x) * (x)`	`(int)SQ(2.0) % 2`
`#define SQ(x) ((x) * (x))`	`SQ(i++)` `SQ(f())`

Arrays

- Arrays are second class citizens

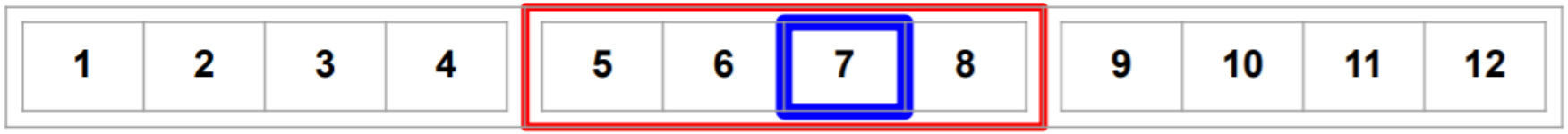
- With an array, you can NOT:
  - Change the size after initialization
  - Assign a new array using ‘=’

- In addition, arrays automatically ‘decay’ into pointers, losing information about their size (with few exceptions).
  - More on array decay next week (after you learn pointers)!

2D Arrays

- Multi-dimensional arrays are typically contiguous.

```
int arr[3][4] = {{1, 2, 3, 4},{5, 6, 7, 8},{9, 10, 11, 12}};
int i = arr[1][2];
```



- They also need additional information to index into the correct position. When passed to a function for example, it needs to know how many values to ‘skip’ to get to an inner array.

Strings

- `long int strlen(const char* source);`
  - Calculates the length of a given string, up to the first null character.
- `char* strcpy(char* destination, const char* source);`
  - Copies the source string to the destination character array.
- `int strcmp (const char* str1, const char* str2);`
  - Compares two strings and returns 0 if both strings are identical.
- `char *strcat(char *dest, const char *src);`
  - Concatenates two strings and stores the result in the first argument.