Style Guide

PART I

Documenting Functions

READ THROUGH THIS ON YOUR OWN!

Some things to remember about Comments

How to add comments

- // ← for a few lines or after a line of code
  - You can select a group of code and ctrl - // to comment out several lines at a time
  - If you ctrl- // on a comment it will uncomment the line
  - This can be useful in debugging – by isolating parts of your code
- Block comments

```
/*
    <anything between these will be commented>
*/
```

Commenting your code

For all programs in this class

Before EVERY FUNCTION

- Use comments to describe your program

Data Table

- The declaration section must contain a data table
- The data table
  - states the use of the variable or named constant and
  - how its value is obtained/used.

Other comments should be used throughout your code to

- Describe what each section is doing
  - (think in terms of input, processing, & output)
- Complicated parts of the code → be descriptive!

Try to line up comments as best as you can!

How to doc your code

First thing in your code should be your name and assignment info

```
/**********************************************************
* AUTHOR    :
* LAB #0    : Template
* CLASS     :
* SECTION   :
* DUE DATE :
***********************************************************/
```

Preprocessor Directives then doc for the main program

Next…

```
#include <iostream>
#include<iomanip>
#include <string>
using namespace std;
/*************************************************************
*
* ADD & MULTIPLY TWO INTS
*____________________________________________________________
* This program does whatever this program does
*  save this template and fill in the appropriate info for
*  your program
*____________________________________________________________
* INPUTS:
*   int1: First integer to be summed received as input
*   int2: Second integer to be summed received as input
*
* OUTPUTS:
*   sum    : the sum of the two ages
*   product: The product of the two integers
*************************************************************/
```

Prototypes

Next

```
/**********************************************************************
 * PrintHeader
 *   This function receives receives an assignment name, type
 *   and number then outputs the appropriate header
 *   - returns  nothing → This will output the class heading.
 *********************************************************************/
void PrintHeader(string asName, // IN - assignment Name
                 char   asType, // IN - assignment type
                                //       (LAB or ASSIGNMENT)
                 int    asNum); // IN - assignment number
```

Next → int main

```
int main ()
{
       // declare your variables here - include your data table

       // PrintHeader - Will output a header for this assignment
       PrintHeader("Functions", 'A', 14);

       // INPUT:  A description of what is being input.

       // PROCESSING:  Detail what is being processed.

       // OUTPUT:  Details of what is being output.
}
```

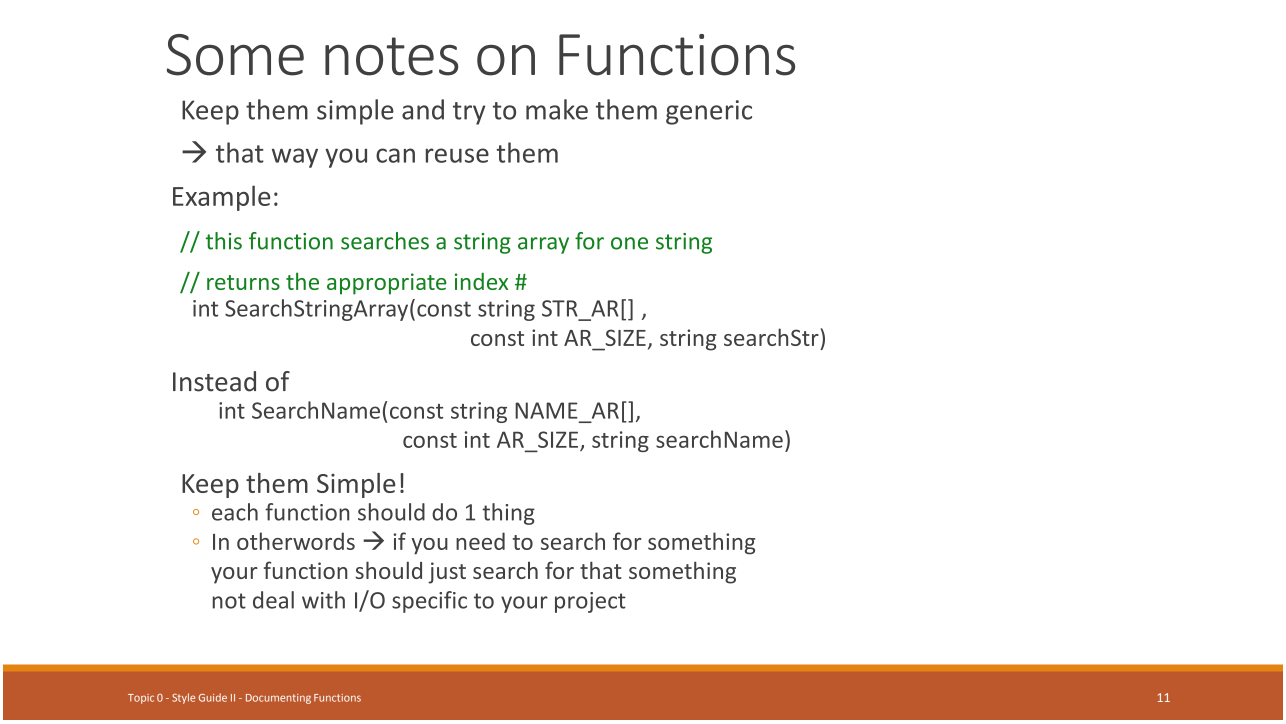
FUNCTIONS should go in another file and should be documented

```
/***********************************************************
 *
 * FUNCTION PrintHeader
 *_________________________________________________________
 * This function receives an assignment name, type
 *   and number then outputs the appropriate header -
 *   returns nothing.
 *_________________________________________________________
 * PRE-CONDITIONS
 *   The following need previously defined values:
 *      asName: Assignment Name
 *      asType: Assignment Type
 *      asNum : Assignment Number
 *
 * POST-CONDITIONS
 *     This function will output the class heading.
 *     <Post-conditions are the changed outputs either
 *      passed by value or by reference OR anything affected
 *      by the function>
 ************************************************************/
 void PrintHeader(string asName, // IN - Assignment Name
                  char   asType, // IN - assignment type
                                 //    - (LAB or ASSIGNMENT)
                  int    asNum)  // IN - assignment number
{
```

Function Definition

```
void PrintHeader(string asName, // IN - assignment Name
                 char    asType, // IN - assignment type
                                 //    - (LAB or ASSIGNMENT)
                 int     asNum   // IN - assignment number
{
  cout << left;
  cout << "**************************************************\n";
  cout << "*  PROGRAMMED BY : Juan Leon\n";
  cout << "*  " << setw(14) << "STUDENT ID" << ": 7502312\n";
  cout <<  "*  " << setw(14) << "CLASS"      << ": CS1B --> MW - 6p-7:30p\n";
  cout << "*  " ;

  // PROC - This will output “LAB #” or “ASSIGNMENT #”  based on the
  //        asType and adjust the setw accordingly
  if (toupper(asType) == 'L')
  {
      cout << "LAB #"        << setw(9);
  }
  else
  {
      cout << "ASSIGNMENT #" << setw(2);
  }
  cout << asNum << ": " << asName << endl;
  cout << "*******************************************\n\n";
  cout << right;
}
```



Some notes on Functions

Keep them simple and try to make them generic

→ that way you can reuse them

Example:

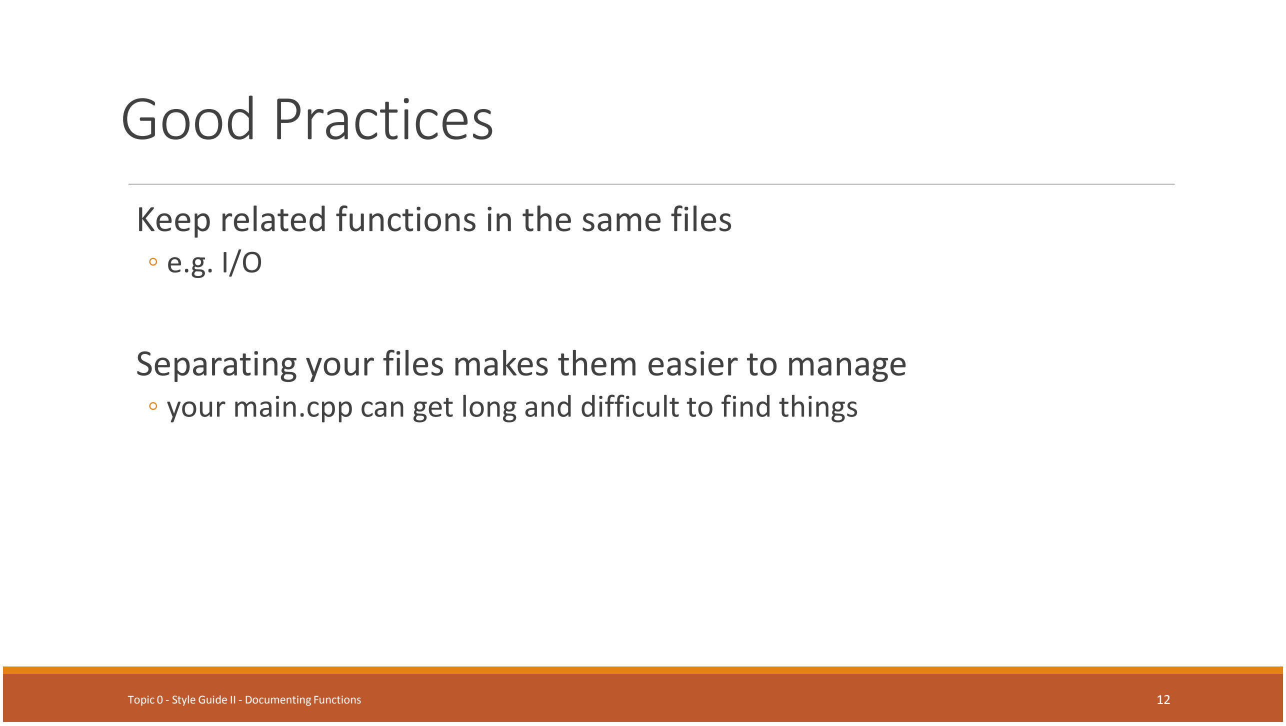
```
// this function searches a string array for one string
// returns the appropriate index #
 int SearchStringArray(const string STR_AR[] ,
                       const int AR_SIZE, string searchStr)
```

Instead of

```
    int SearchName(const string NAME_AR[],
                   const int AR_SIZE, string searchName)
```

Keep them Simple!

- each function should do 1 thing
- In otherwords → if you need to search for something your function should just search for that something not deal with I/O specific to your project



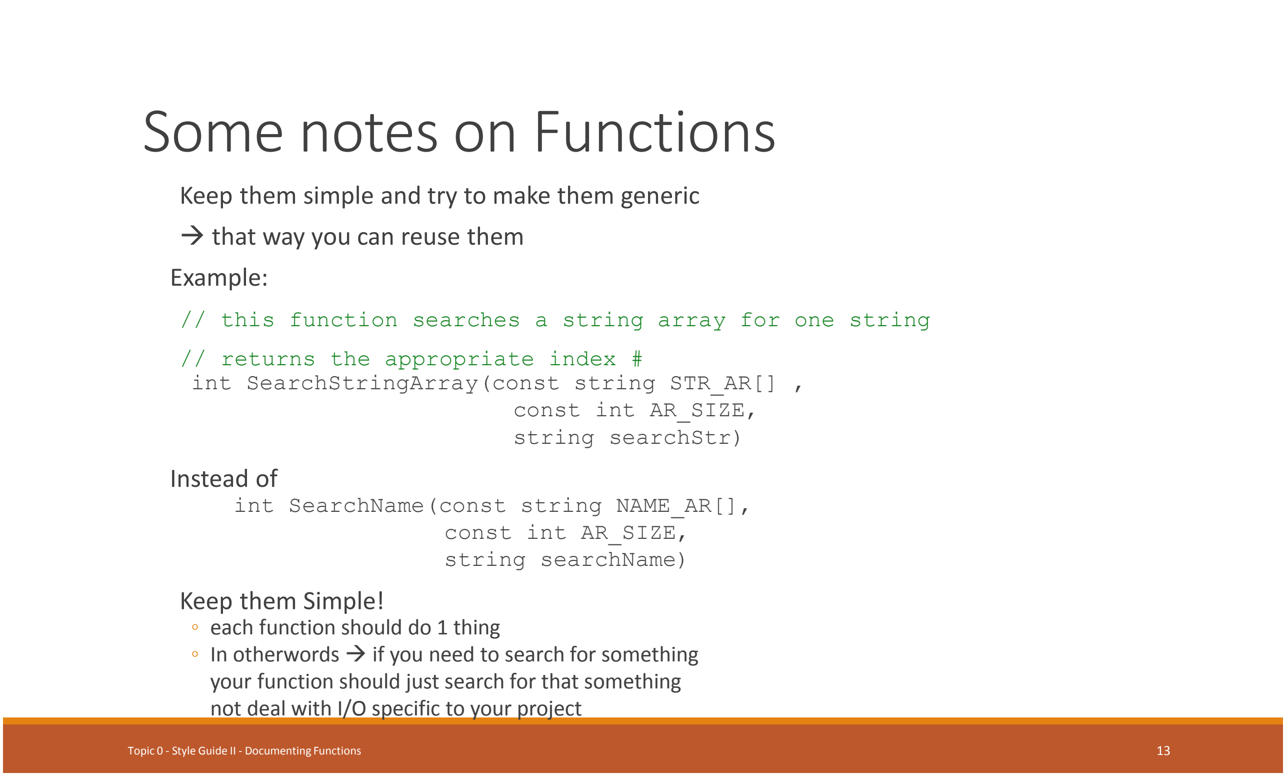
Good Practices

Keep related functions in the same files

- e.g. I/O

Separating your files makes them easier to manage

- your main.cpp can get long and difficult to find things



Some notes on Functions

Keep them simple and try to make them generic

→ that way you can reuse them

Example:

```
// this function searches a string array for one string
// returns the appropriate index #
 int SearchStringArray(const string STR_AR[] ,
                        const int AR_SIZE,
                        string searchStr)
```

Instead of

```
    int SearchName(const string NAME_AR[],
                   const int AR_SIZE,
                   string searchName)
```

Keep them Simple!

- each function should do 1 thing
- In otherwords → if you need to search for something your function should just search for that something not deal with I/O specific to your project