Components of a Program

Class 3

File Organization

from the style guide:

- each C++ program is a collection of one or more source files
- organize the material in each file as follows
  1. a comment explaining the purpose of this file, along with your full name
  2. preprocessor typedefs and defines, if any
  3. preprocessor includes, if any
  4. a namespace statement, if used
  5. function headers, if any
  6. the main function, if present
  7. other functions, if any

Order of a C++ Program

```
// Jane Doe
// A program to print a greeting
#include <iostream>
using namespace std;
int main()
{
 cout << "Hello, world!" << endl;
 return 0;
}
```

comment: your name

explanation

preprocessor directive

namespace statement

the function main

cout

- a variable pre-defined in the iostream library
- console output
- used with the stream insertion operator << to send output to the terminal screen

```
cout << "Hello, world";
```

- can be use to send more than one thing to the screen via chaining

```
cout << "Hello " << "world";
```

- to end a line of output, use endl

```
cout << "Hello world" << endl;
```

example programs 2-4 – 2-6 on pages 33 – 34

Escape Sequences

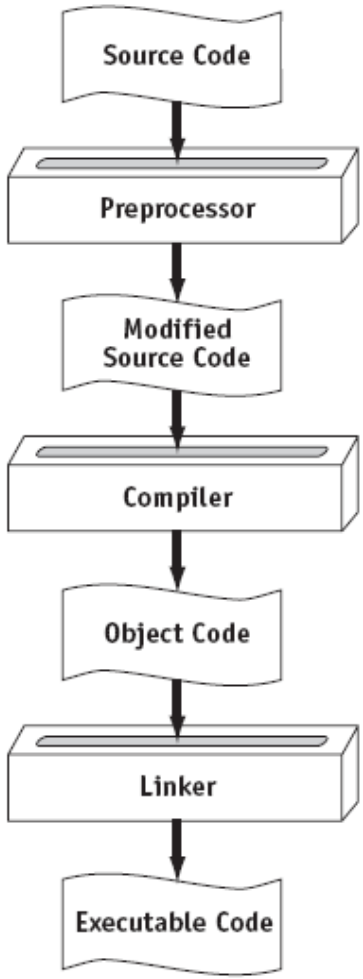
- program 2-6 uses escape sequences
- table on page 35

Sequence	Name
`\n`	newline
`\t`	tab
`\\`	backslash
`\"`	double quote

The #include Directive

```
#include <iostream>
```

- not a C++ language statement
- a command handled by the preprocessor
- inserts the contents of the system iostream library into the C++ source code
- note there is no semicolon at the end of the line because this is not a C++ language statement



Literal Values

- a program can contain literal values
  `apples = 20;` an integer literal
  `pi = 3.1415;` a floating point literal
  `cout << "Hello world";` a string literal
- a literal is a value that is literally written in the code
- as opposed to a value that comes from somewhere else:
  `cin >> hours;`

Variables

Variable

a programmer-defined **named storage location** in memory for holding a piece of data

- must be declared
- must be of a specific type

Identifiers

- a programmer-defined name for some part of a program
  - variables
  - functions
- a name should represent the purpose of the thing being named
- variables represent things and so should be nouns

  `number_of_apples`

  `hours`
- functions represent actions and so should be verbs

  `calculate_pay`

  `deduct_interest`

Identifier Syntax Rules

- must begin with a letter
- variables and functions should contain only lower case letter
- may contain letters, digits, underscores; no other characters
- case sensitive (but variables and functions should not have upper case letters)
- cannot be a reserved word

Identifier	**Valid?**	**Notes**
`total_sales`	Yes	good
`totalSales`	Yes	bad style
`total.sales`	No	period illegal
`sales_4th_quarter`	Yes	
`4th_quarter_sales`	No	leading digit illegal
`total$`	No	$ illegal
`namespace`	No	reserved word