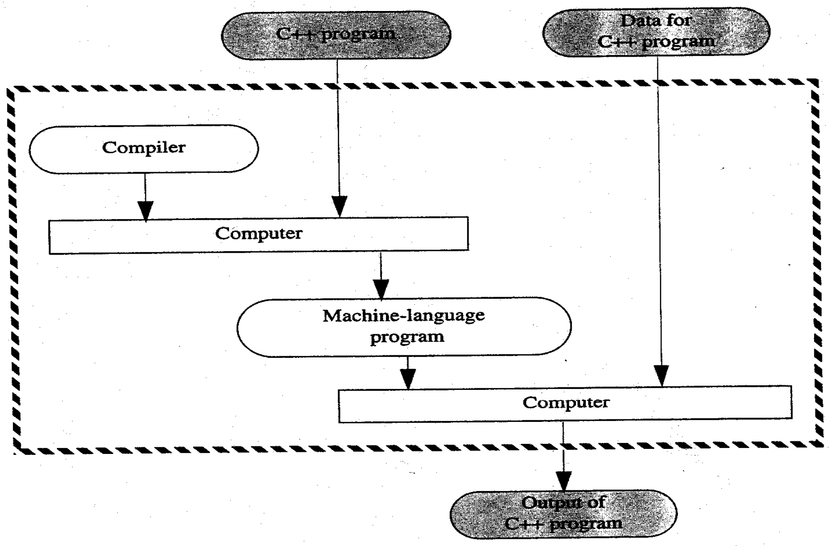
What we will learn about this week:

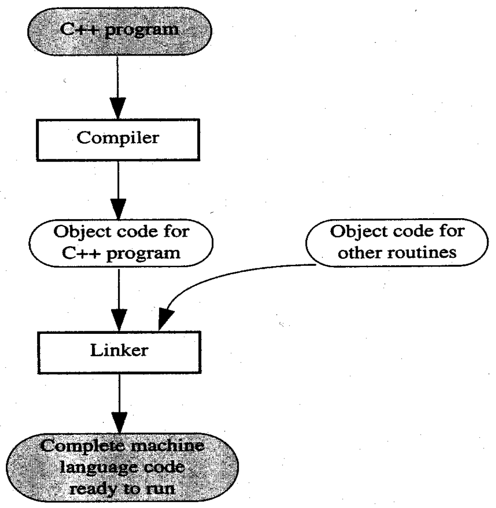
- Types of errors
- Variables
- Data types

Display 1.4 Compiling and Running a C++ Program (Basic Outline)



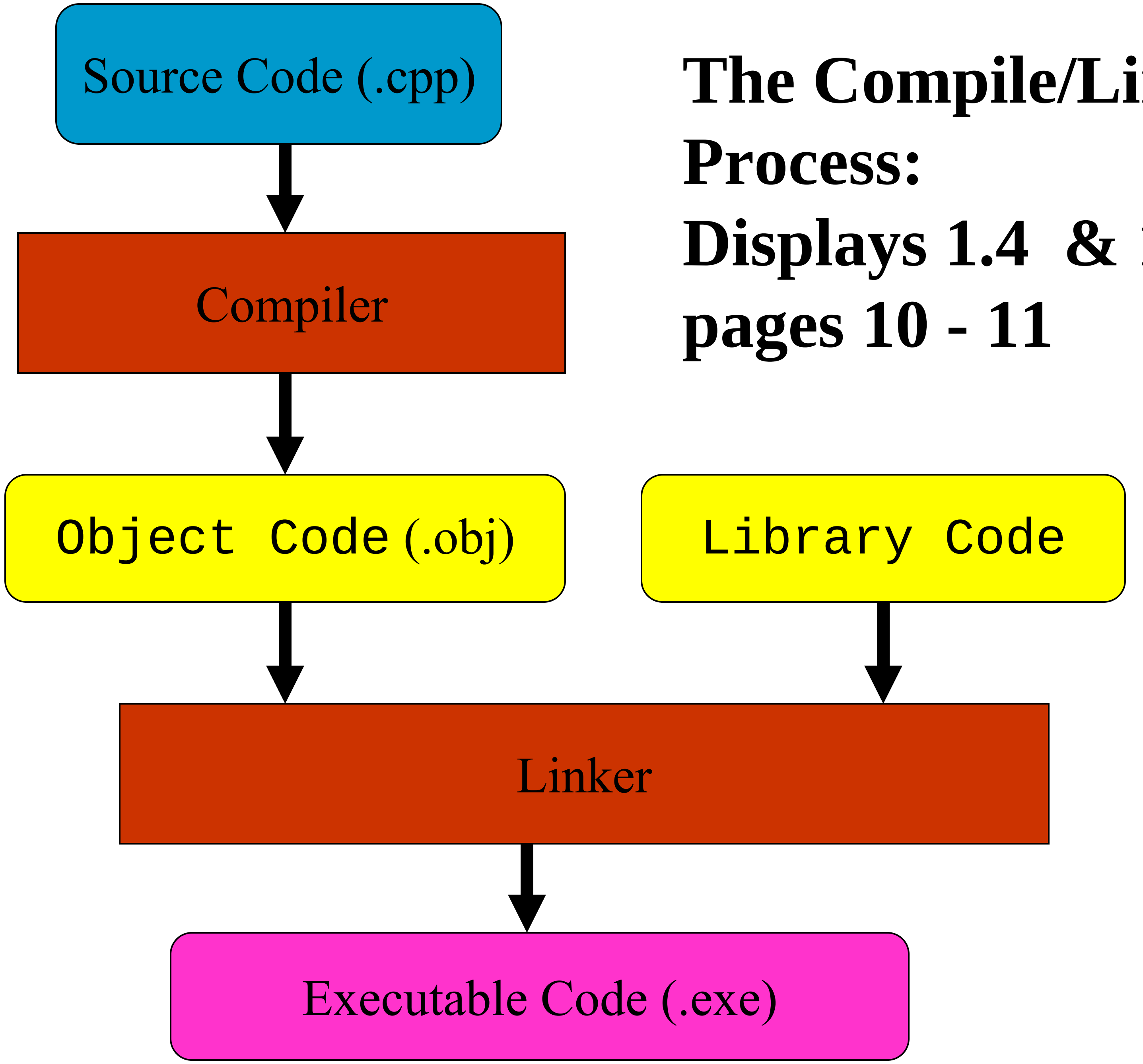
First Compile

Display 1.5 Preparing a C++ Program for Running

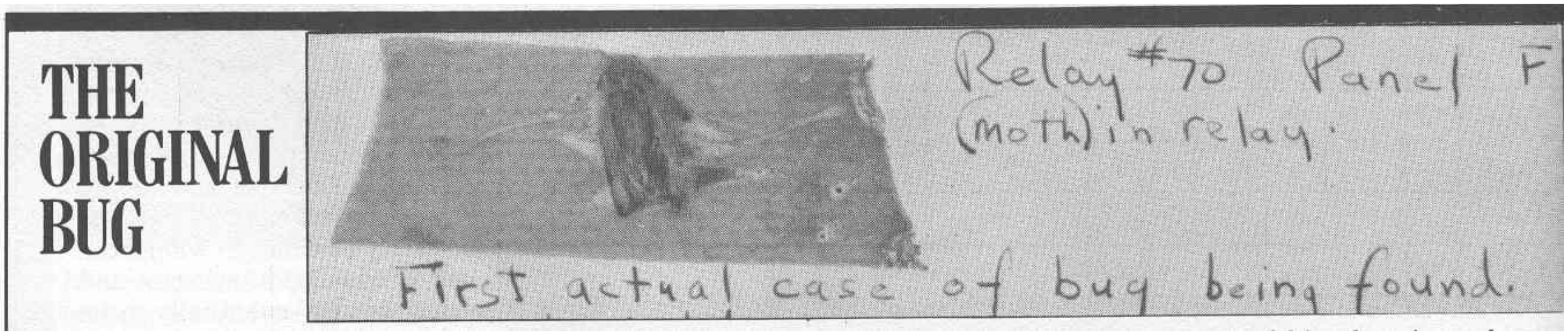

C++ program
Compiler
Object code for C++ program
Object code for other routines
Linker
Complete machine language code ready to run

Then Link

The Compile/Link Process:
Displays 1.4 & 1.5 pages 10 - 11



Moth crushed in relay of Mark II computer


THE ORIGINAL BUG

Relay #70 Panel F (moth) in relay.

First actual case of bug being found.

Entered into the logbook of
Commodore Grace Hopper
(Mother of Cobol)

Types of errors

- **design errors** -- if you solved the wrong problem, you have design errors.
- **syntax errors** -- violation of language’s grammar rules, usually caught by the compiler, and reported by **compiler error messages.**
- **run-time errors** -- a program that compiles may die while running with a run-time error message that may or may not be useful.
- **logic error** -- a program that compiles and runs to normal completion may not do what you want. May be a design error.

Debugging

Address errors one at a time:

- Start with first error (all others could be cascade errors)
- Error might be on previous line
- Watch spelling and semicolons
- Watch spacing on output
- Warnings Vs. Errors
- **Syntax** errors (like omitting a “;”) vs. **Runtime** errors (trying to divide by 0)
- (Syntax basically means rules of grammar)

  Syntax errors caught in the compiling process
- Run-time errors caught when the program is run
- **Logic** errors are not caught by the computer (using a + instead of a *)

Layout of simple C++ program

#include <iostream>	An include directive (include iostream library)
using namespace std;	Use the standard collection of names
int main()	Declares main function
{	Start of main's body
variable _declarations	Declaration/initialization of variables
Statement1;	Body of main
Statement2;	
…	
Statement_last;	
return 0;	Says "end program here"
}	End of main's body

```
#include <iostream>
using namespace std;

int main( )
{
   int number_of_pods, peas_per_pod, total_peas;

   cout << "Press return after entering a number.\n";
   cout << "Enter the number of pods:\n";
   cin >> number_of_pods;
   cout << "Enter the number of peas in a pod:\n";
   cin >> peas_per_pod;

   total_peas = number_of_pods * peas_per_pod;

   cout << "If you have ";
   cout << number_of_pods;
   cout << " pea pods\n";
   cout << "and ";
   cout << peas_per_pod;
   cout << " peas in each pod, then\n";
   cout << "you have ";
   cout << total_peas;
   cout << " peas in all the pods.\n";

   return 0;
}
```

Note indentation and spacing

Sample Output

```
Press return after entering a number.
Enter the number of pods:
10
Enter the number of peas in a pod:
9
If you have 10 pea pods
and 9 peas in each pod then
you have 90 peas in all of the pods.
```

Sample Student Output

```
Press return after entering a number.
Enter the number of pods:
10
Enter the number of peas in a pod:
6
If you have 10pea pods
and 6peas in each pod, then
you have 60 peas in all the pods.
Press any key to continue
```

Take a Break!

Identifiers

```
int number_of_pods, peas_per_pod, total_peas;
```

- The program manipulates data

- A variable sets aside memory for data and gives that place a name.

- The name of a variable (or anything else in C++ that needs naming) is called an **identifier.**

- Always has a value (even if garbage)

Valid / Invalid Identifiers

- Always start with a letter or underscore
- Composed of letters, numbers and underscores.
- Case sensitive (a not equal to A)
- Avoid reserved words (**appendix 1)**
- Use descriptive names

- Valid Examples:

x, _abc, A_2b, ThisIsAVeryLongIdentifier

- Invalid Examples:

12, 3X, %change, myFirst.c, data-1

Avoid reserved words

Appendix 1

asm	do	inline	return	typedef
auto	double	int	short	typeid
bool	dynamic_cast	log	signed	typename
break	else	long	sizeof	union
case	enum	mutable	static	unsigned
catch	explicit	namespace	static_cast	using
char	extern	new	struct	virtual
class	false	operator	switch	void
const	float	private	template	volatile
const_cast	for	protected	this	wchar_t
continue	friend	public	throw	while
default	goto	register	true	
delete	if	reinterpret_cast	try	

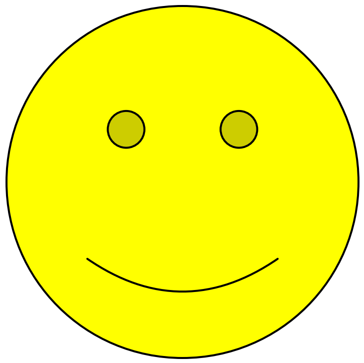
Variable names should be meaningful

Good:

distance

height

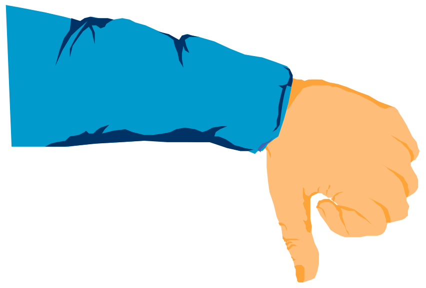
dimes



Bad:

var1

x



Variable Declaration

- Every variable in C++ must be *declared*.
- A declaration introduces a name to the compiler and specifies the type of data that is to be stored in the variable.
- A variable declaration has the form

  Type_name variable_name1, variable_name2, ***…*** *;*
- The kind of data held in a variable is it **type.**
- You ***must*** declare all variables prior to use.

Type_name Variable_name_1, Variable _name_2;

Examples:

int number_of_bars;

double distance, time, speed;

Initializing variables

All variables have a value even if it is garbage. It is generally a good idea to initialize your variables.

Examples:

```
int count=0, limit=10;
OR
int count(0), limit(10);
```

Beware of un-initialized variables!

Data Types

- Integer
- Float
- Char

You must decide the appropriate type of data to use:

What would you use to keep tract of how many apples you have?

What would you use to keep tract of how many pounds of applesauce?

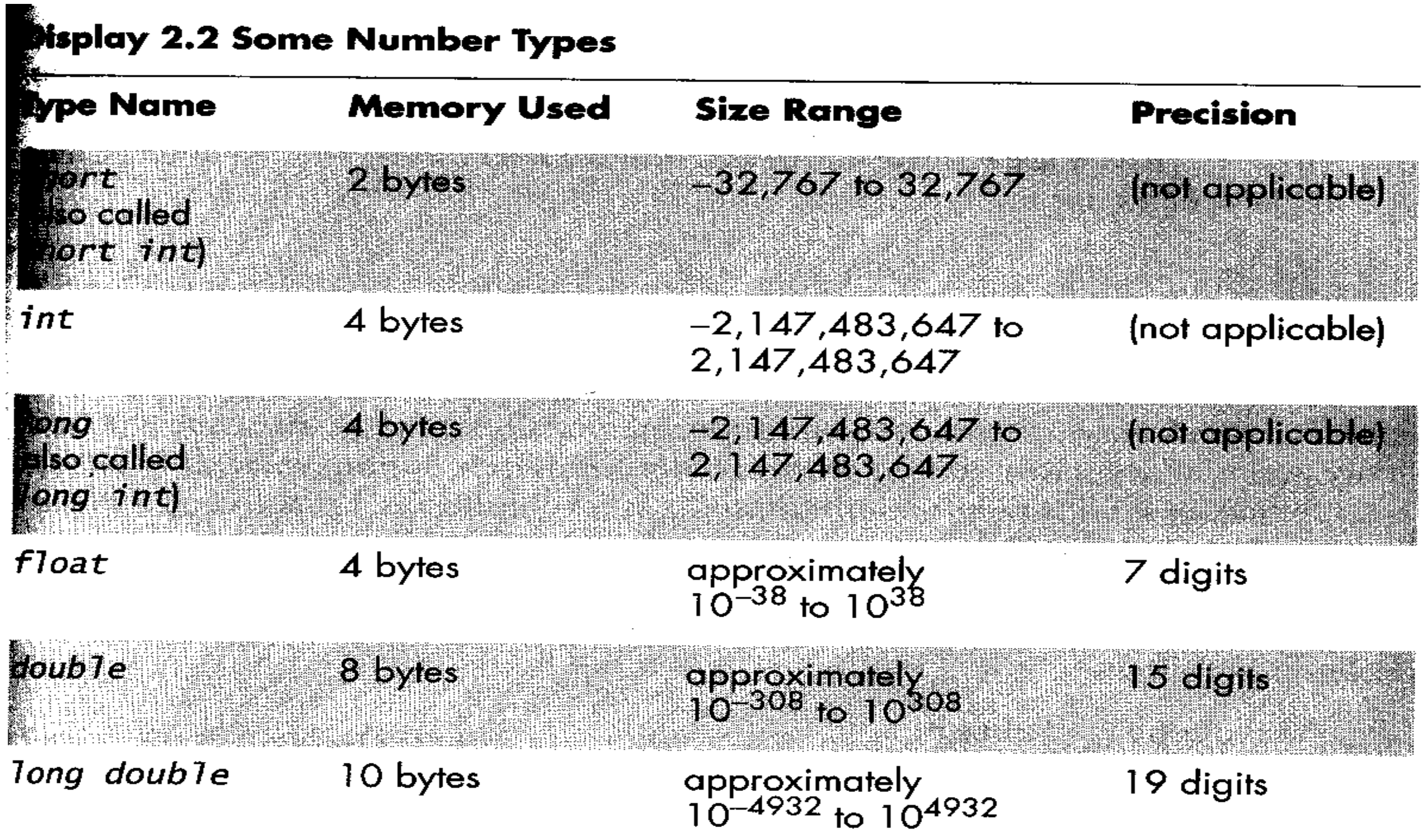
What would you use for money?

Numeric Data Types
(May be different with a different compiler)

Type	Memory	Size
short	2 bytes	-32,767 to 32, 767
int	4 bytes	-2,147,483,647 to 2,147,483,647
long	4 bytes	-2,147,483,647 to 2,147,483,647
float	4 bytes	~10^{-38} to 10^{38} (7 digits)
double	8 bytes	~10^{-308} to 10^{308} (15 digits)
long double	10 bytes	~10^{-4932} to 10^{4932} (19 digits)

Number Types
2 is not the same as 2.0

(Display 2.2 page 57)



isplay 2.2 Some Number Types

ype Name	Memory Used	Size Range	Precision
ort lso called *ort int*)	2 bytes	–32,767 to 32,767	(not applicable)
int	4 bytes	–2,147,483,647 to 2,147,483,647	(not applicable)
ong lso called *ong int*)	4 bytes	–2,147,483,647 to 2,147,483,647	(not applicable)
float	4 bytes	approximately 10^{-38} to 10^{38}	7 digits
double	8 bytes	approximately 10^{-308} to 10^{308}	15 digits
long double	10 bytes	approximately 10^{-4932} to 10^{4932}	19 digits

Char and Bool

char **is a special type that is designed to hold single members of the ASCII character set.**

- ***Uppercase and Lowercase not the same!***
- ***‘A’ <> ‘a’***
- ***“A” is not the same as ‘A’***
- ***char nine = ‘9’; is not the same as int nine = 9;***
- ***char nine = ‘9’; is not the same as char nine = 9;***

bool **has only two values:** ***true*** **and** ***false*****.**

Constants

- Same as a variable except word the keyword const in front.

```
const double PI = 3.14159;
```

- Must have a value.
- Generally done all in upper case.
- You cannot change a constant while the program is running.

Assignment Statements

In an assignment statement, the right hand side expression is evaluated, then the variable on the left hand side is set to this value.

Syntax: variable = expression;

Setting the value of variables:

```
num_of_bars = 37;
total_weight = one_weight;
num_of_bars = num_of_bars + 3;
total_weight = one_weight * num_of_bars
```

Arithmetic Operators

```
total_peas = number_of_pods * peas_per_pod;
```

- The asterisk, *, is used for multiplication.
- This line multiplies the already entered values of number_of_pods and peas_per_pod to give a value which is stored (assigned to) total_peas.

Common arithmetic operators are encoded:

Addition	**+**	**Multiplication**	*****
Subtraction	**-**	**Division**	**/**

Precedence

When two operators appear in an arithmetic expression, there are PRECEDENCE rules that tell us what happens.

Evaluate the expression,

X + Y * Z

by multiplying Y and Z first then the result is added to X.

Rule: Do inside most parentheses first, then

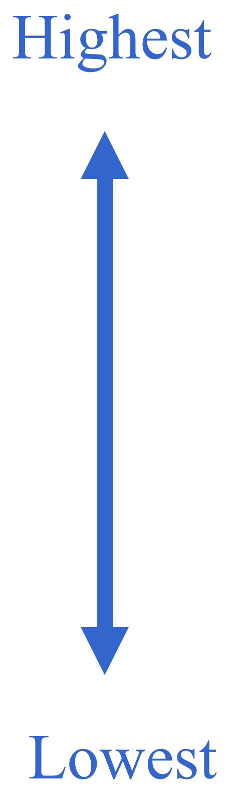
multiplication and division next,

additions and subtractions next, and

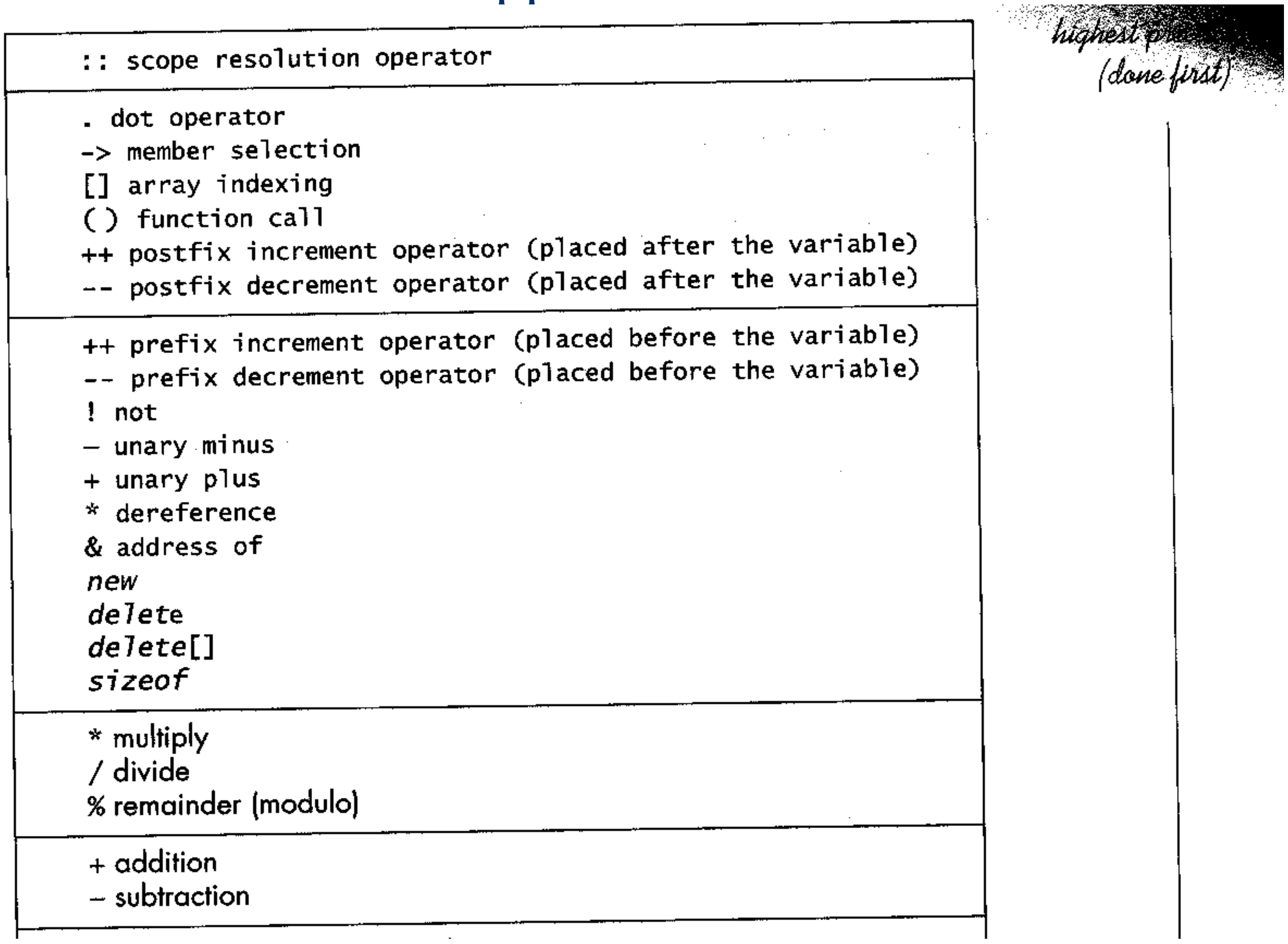
break all ties left to right.

Precedence Rules

Unary operators	+, -, ++, --, and !
Arithmetic operators	*, /, %
Arithmetic operators	+, -
Boolean operators	<, >, <=, >=
Boolean operators	==, !=
Boolean operators	&&
Boolean operators	||



Precedence
Appendix 2



`::` scope resolution operator
`.` dot operator `->` member selection `[]` array indexing `( )` function call `++` postfix increment operator (placed after the variable) `--` postfix decrement operator (placed after the variable)
`++` prefix increment operator (placed before the variable) `--` prefix decrement operator (placed before the variable) `!` not `–` unary minus `+` unary plus `*` dereference `&` address of *new* *delete* *delete*[] *sizeof*
* multiply / divide % remainder (modulo)
+ addition – subtraction

highest p
(done first)

Reading Assignment

Section 2.4 pages 46-54

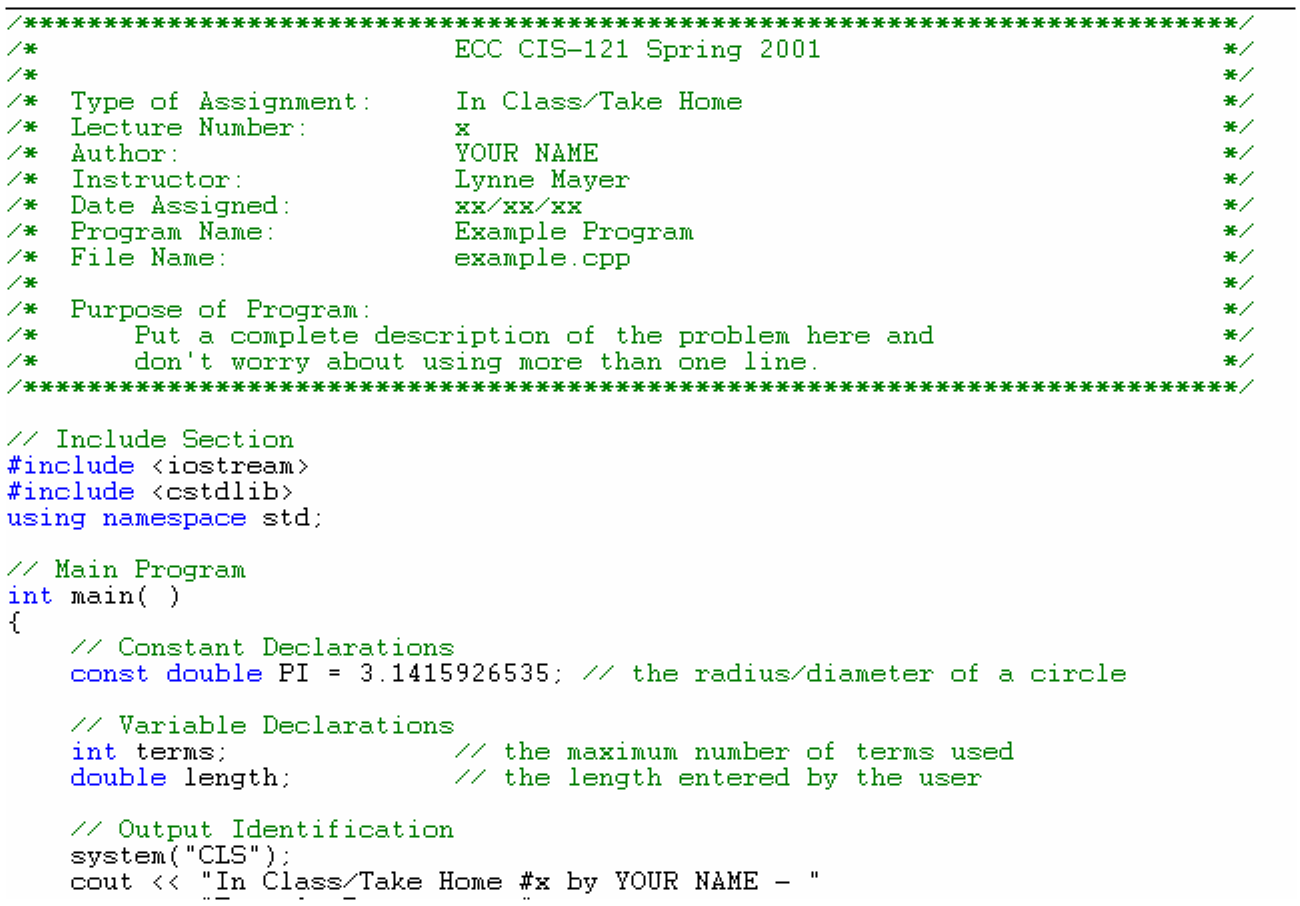
Section 5.2 pages 216 - 231

Take a Break!

Comments

- //
- /* */
- End of line comments
- Single line comment
- Block Comments

The Standard Header for this class



```
/*****************************************************************************/
/*                            ECC CIS-121 Spring 2001                       */
/*                                                                           */
/*   Type of Assignment:      In Class/Take Home                             */
/*   Lecture Number:          x                                              */
/*   Author:                  YOUR NAME                                      */
/*   Instructor:              Lynne Mayer                                    */
/*   Date Assigned:           xx/xx/xx                                       */
/*   Program Name:            Example Program                                */
/*   File Name:               example.cpp                                    */
/*                                                                           */
/*   Purpose of Program:                                                     */
/*       Put a complete description of the problem here and                  */
/*       don't worry about using more than one line.                         */
/*****************************************************************************/

// Include Section
#include <iostream>
#include <cstdlib>
using namespace std;

// Main Program
int main( )
{
    // Constant Declarations
    const double PI = 3.1415926535; // the radius/diameter of a circle

    // Variable Declarations
    int terms;              // the maximum number of terms used
    double length;          // the length entered by the user

    // Output Identification
    system("CLS");
    cout << "In Class/Take Home #x by YOUR NAME - "
```

Go To Lmayer in the instructor folders (drive I) found under “My Computer”. Then open the CIS121 folder.

TempConvertCtoF Problem Description:

Write a program that converts a temperature in degrees Celsius to the corresponding temperature in degrees Fahrenheit. The program must read in a temperature (in degrees Celsius) from the console. The program must display the converted temperature to the console. The program should handle the temperatures as floating point numbers (doubles).

Note: The following equation is used to convert Degrees C to Degrees F:

Deg_F = Deg_C * 9.0 / 5.0 + 32

Example Output:

```
The TempConvertCtoF program by Lynne Mayer
Please enter the temperature in Deg CF: 22
That is 71.6 Degrees Celsius.
```

Required Test Cases:

22
0
100
233

Skills:

●Var ●Con I/O ○Format ○Logic ○Loops ○Functions ○Call by Ref ○File I/O ○Arrays ○Strings ○GM