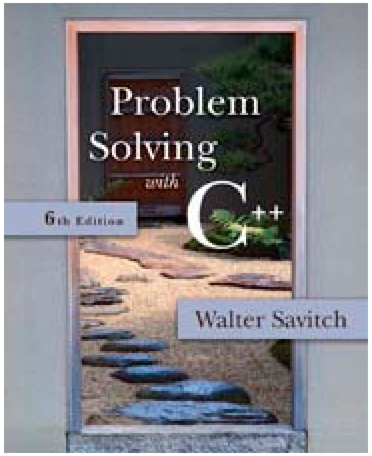
APS105: Lecture 3

Wael Aboelsaadat

wael@cs.toronto.edu

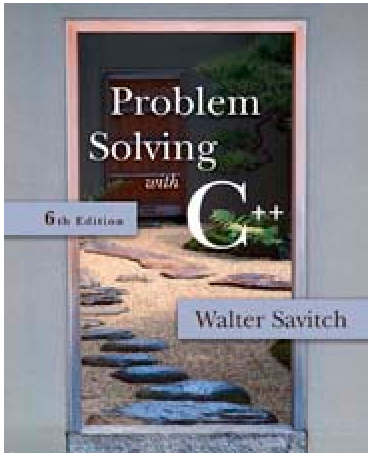
http://ccnet3.utoronto.ca/20079/aps105h1f/

Acknowledgement: These slides are a modified version of the text book slides as supplied by Addison Wesley

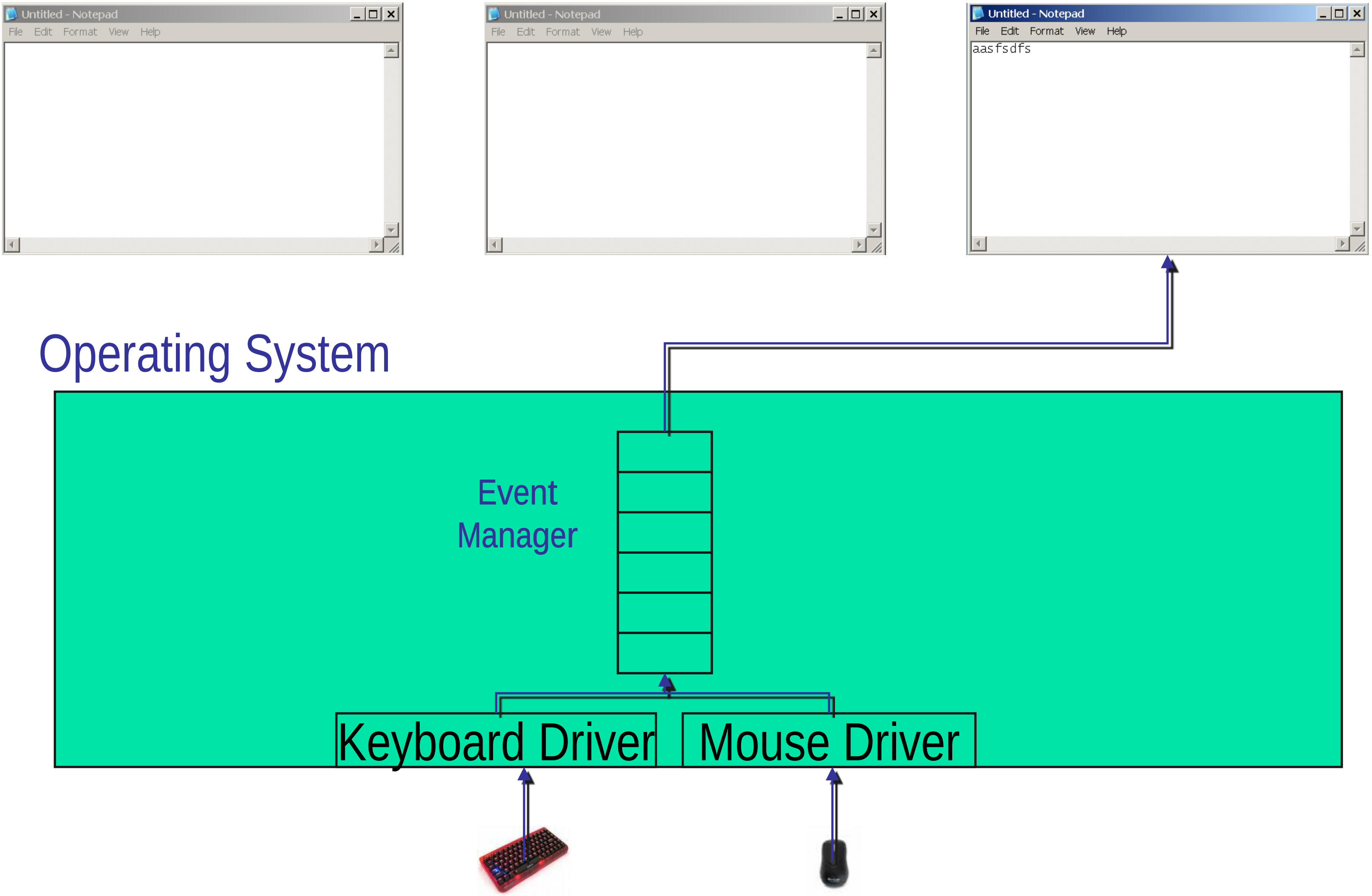


Chapter 1

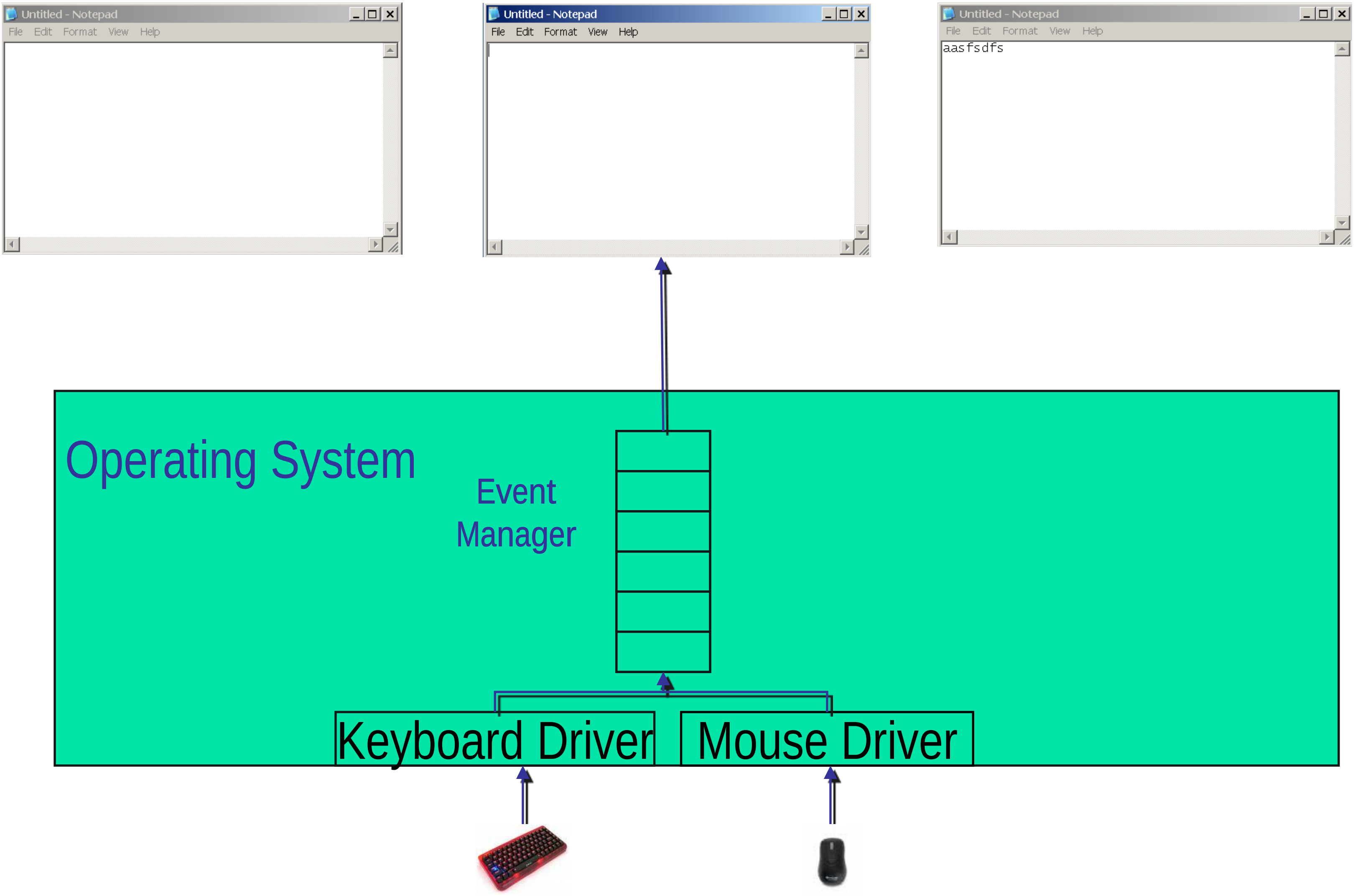
Introduction to Computers and C++ Programming



Operating System



Operating System



Operating System

Event Manager

Keyboard Driver

Mouse Driver

Untitled - Notepad
File Edit Format View Help
Untitled - Notepad
File Edit Format View Help
Untitled - Notepad
File Edit Format View Help
aasfsdfs

Computer Input

- Computer input consists of
    - A program
    - Some data

Display 1.3

High-level Languages

- Common programming languages include …

  C C++ Java Pascal Visual Basic FORTRAN
  COBOL Lisp Scheme Ada

- These high – level languages
  - Resemble human languages
  - Are designed to be easy to read and write
  - Use more complicated instructions than the CPU can follow
  - Must be translated to zeros and ones for the CPU to execute a program

Low-level Languages

- An assembly language command such as

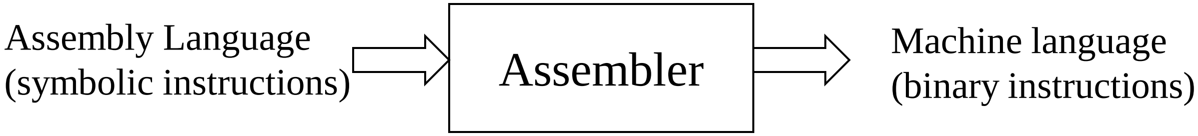
  ADD X Y Z

  might mean add the values found at x and y in memory, and store the result in location z.

- Assembly language must be translated to machine language (zeros and ones)

  0110 1001 1010 1011

- The CPU can follow machine language



Assembly Language
(symbolic instructions)

Assembler

Machine language
(binary instructions)

Compilers

- Translate high-level language to machine language
  - Source code
    - The original program in a high level language
  - Object code
    - The translated version in machine language

Display 1.4

Compilers

C++

```
int main() {
  int nIndex,nSum;
  for( nIndex=0; nIndex<10;nIndex++)
      nSum =+ 2 * nIndex;
}
```

Assembly

```
 .file   "foo.c"
                .text
                .p2align 4,,15
.globl main
                .type   main, @function
main:           push   BP
                mov    $9, AX
                mov    SP, BP
                sub    $8, SP
                and    $-16, SP
                .p2align 4,,15
.L6:            dec    AX
                jns    .L6
                mov    BP, SP
                pop    BP
                ret
                .size   main, .-main
                .ident  "GCC: (GNU) 3.3.1"
```

1s and 0s

```
01010101010001
10101010101111
10101001010101
10010101001000
00000001101111
00000000000000
11111111100001
```

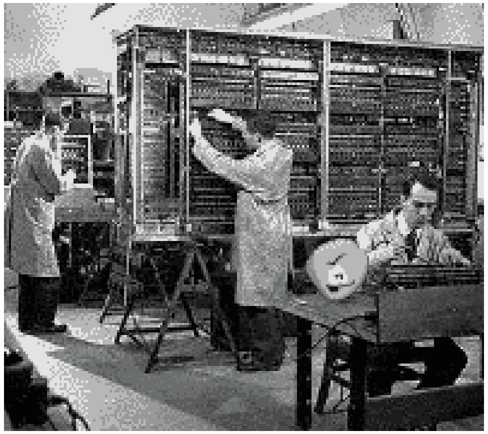
Linkers

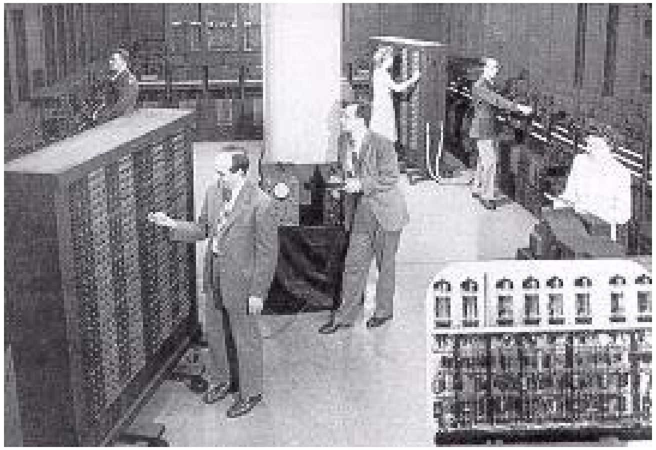
- Some programs we use are already compiled
  - Their object code is available for us to use
  - For example: Input and output routines

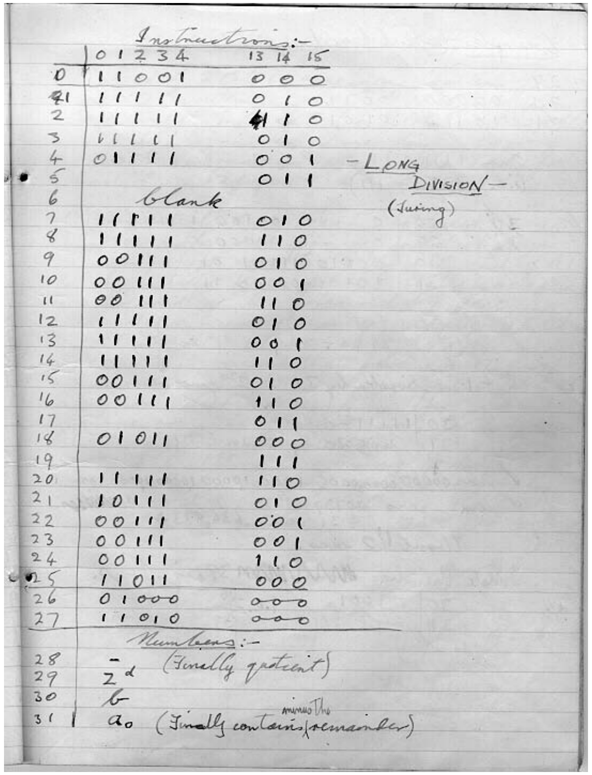
- A Linker combines
  - The object code for the programs we write
    and
  - The object code for the pre-compiled routines
    into
  - The machine language program the CPU can run

Display 1.5

History Note







Instructions:–

	0 1 2 3 4	13 14 15
0	1 1 0 0 1	0 0 0
1	1 1 1 1 1	0 1 0
2	1 1 1 1 1	1 1 0
3	1 1 1 1 1	0 1 0
4	0 1 1 1 1	0 0 1
5		0 1 1
6	blank	
7	1 1 1 1 1	0 1 0
8	1 1 1 1 1	1 1 0
9	0 0 1 1 1	0 1 0
10	0 0 1 1 1	0 0 1
11	0 0 1 1 1	1 1 0
12	1 1 1 1 1	0 1 0
13	1 1 1 1 1	0 0 1
14	1 1 1 1 1	1 1 0
15	0 0 1 1 1	0 1 0
16	0 0 1 1 1	1 1 0
17		0 1 1
18	0 1 0 1 1	0 0 0
19		1 1 1
20	1 1 1 1 1	1 1 0
21	1 0 1 1 1	0 1 0
22	0 0 1 1 1	0 0 1
23	0 0 1 1 1	0 0 1
24	0 0 1 1 1	1 1 0
25	1 1 0 1 1	0 0 0
26	0 1 0 0 0	0 0 0
27	1 1 0 1 0	0 0 0

– LONG DIVISION –

(Turing)

Numbers:–

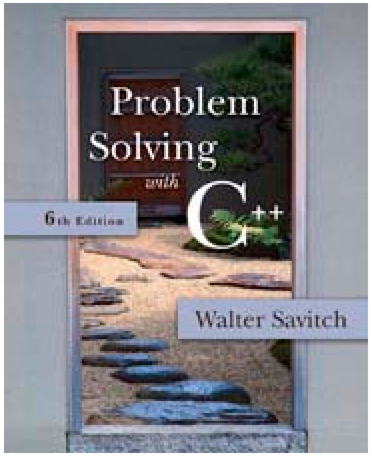
28	–
29	z^d (Finally quotient)
30	b
31	a_0 (Finally contains minus the remainder)

Section 1.1 Conclusion

- Can you…
  - List the five main components of a computer?
  - List the data for a program that adds two numbers?
  - Describe the work of a compiler?
  - Define source code? Define object code?
  - Describe the purpose of the operating system?

1.2

Programming and Problem-Solving



Algorithms

- Algorithm
  - A sequence of precise instructions which leads to a solution
- Program
  - An algorithm expressed in a language the computer can understand

Display 1.6

Program Design

- Programming is a creative process
  - No complete set of rules for creating a program

- Program Design Process
  - Problem Solving Phase
    - Result is an algorithm that solves the problem
  - Implementation Phase
    - Result is the algorithm translated into a programming
language

Problem Solving Phase

- Be certain the task is completely specified
  - What is the input?
  - What information is in the output?
  - How is the output organized?

- Develop the algorithm before implementation
  - Experience shows this saves time in getting your program to run.
  - Test the algorithm for correctness

Implementation Phase

- Translate the algorithm into a programming language
  - Easier as you gain experience with the language
- Compile the source code
  - Locates errors in using the programming language
- Run the program on sample data
  - Verify correctness of results
- Results may require modification of the algorithm and program

Display 1.7

Object Oriented Programming

- Abbreviated OOP

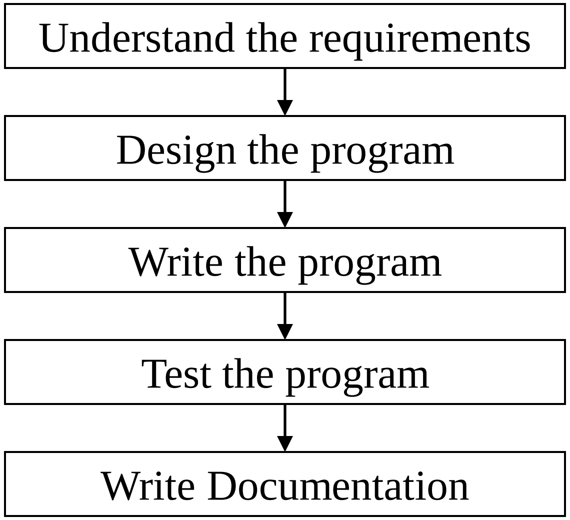
- Used for many modern programs

- Program is viewed as interacting objects
  - Each object contains algorithms to describe its behavior
  - Program design phase involves designing objects and their algorithms

Software Life Cycle

- Analysis and specification of the task (problem definition)
- Design of the software (object and algorithm design)
- Implementation (coding)
- Maintenance and evolution of the system
- Obsolescence

Software house: what happens inside?



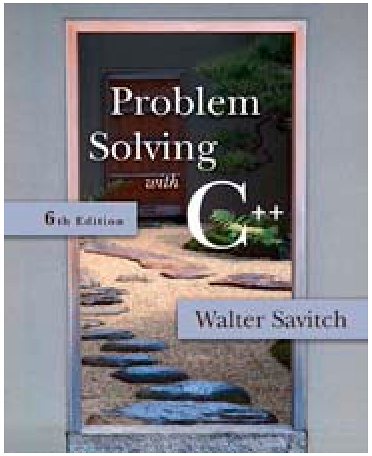
Understand the requirements	Project Managers
Design the program	Software Architects
Write the program	Development/Programmers
Test the program	Quality Assurance
Write Documentation	Documentation

Section 1.2 Conclusion

- Can you…
  - Describe the first step to take when creating a program?
  - List the two main phases of the program design process?
  - Explain the importance of the problem-solving phase?
  - List the steps in the software life cycle?

1.3

Introduction to C++



Introduction to C++

- Where did C++ come from?
  - Derived from the C language
  - C was derived from the B language
  - B was derived from the BCPL language

- Why the ‘++’?
  - ++ is an operator in C++ and results in a cute pun

C++ History

- C developed by Dennis Ritchie at AT&T Bell Labs in the 1970s.
  - Used to maintain UNIX systems
  - Many commercial applications written in c
- C++ developed by Bjarne Stroustrup at AT&T Bell Labs in the 1980s.
  - Overcame several shortcomings of C
  - Incorporated object oriented programming
  - C remains a subset of C++

A Sample C++ Program

- A simple C++ program begins this way

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

int main()
{
```

- And ends this way

```
        return 0;
}
```

Display 1.8

Explanation of code (1/5)

- Variable declaration line

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

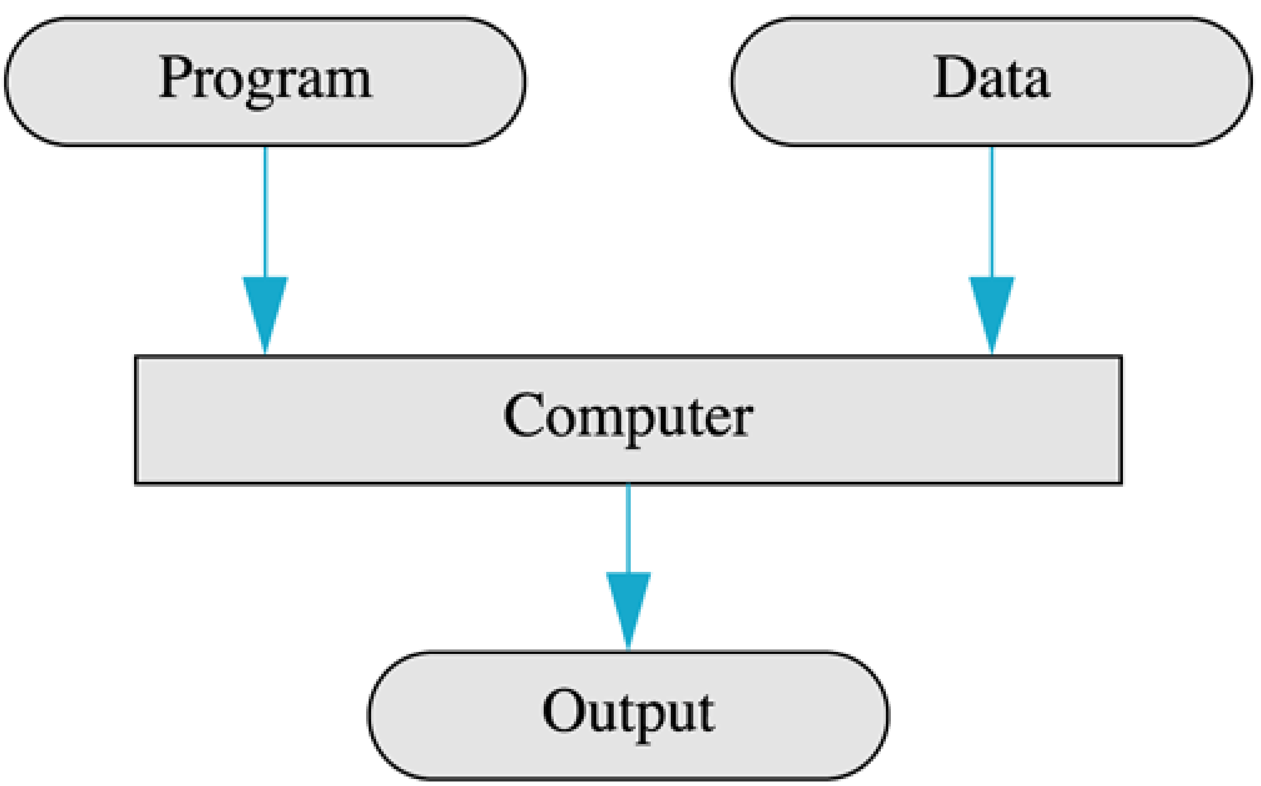
  - Identifies names of three variables to name numbers
  - int means that the variables represent integers

Display 1.3

Back

Next

Simple View of Running a Program

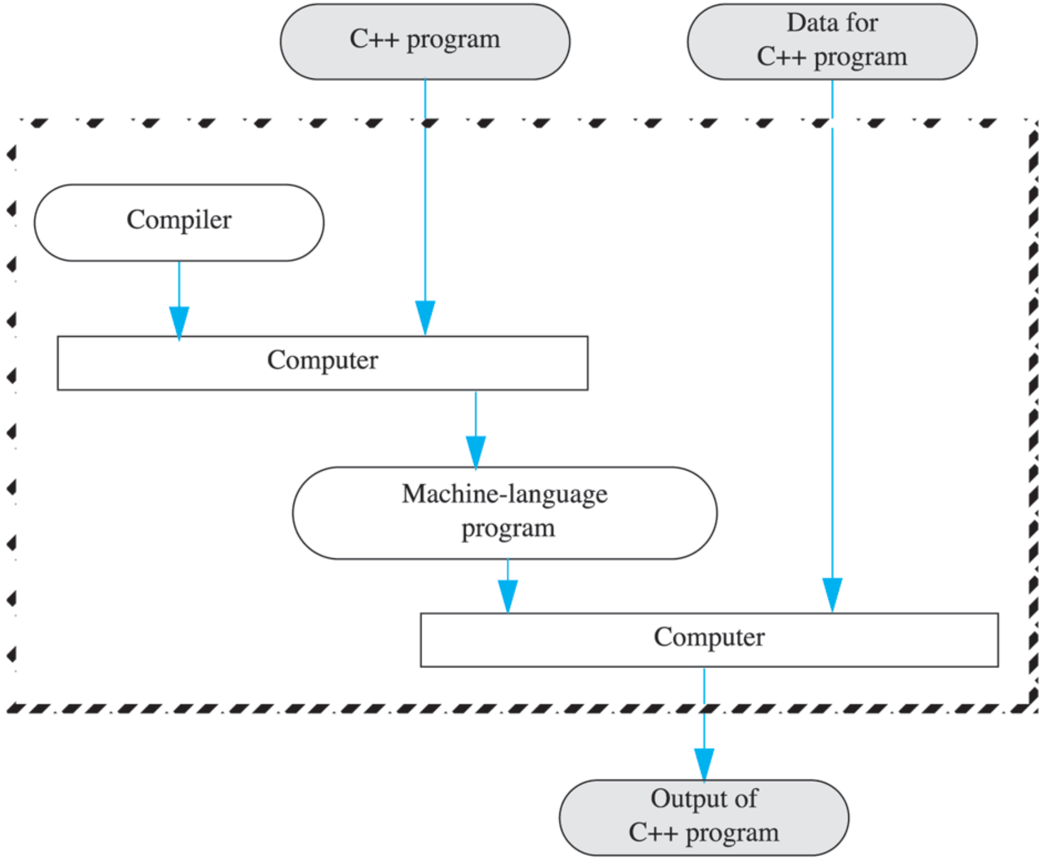


Display 1.4

Back

Next

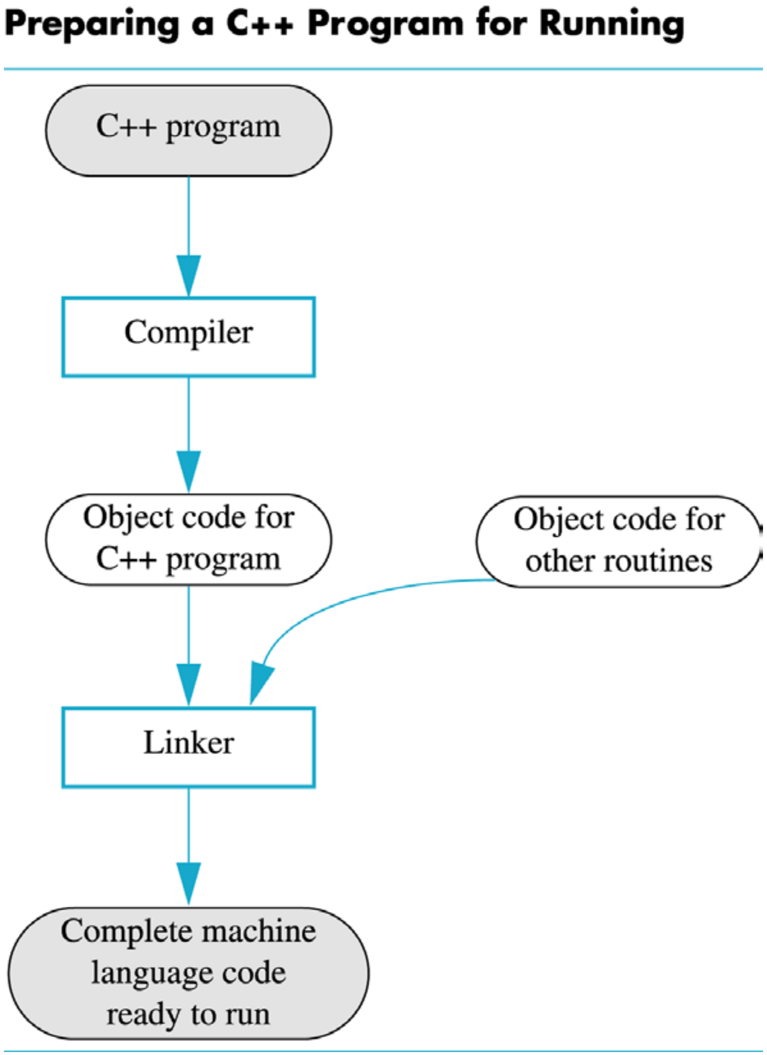
Compiling and Running a C++ Program (Basic Outline)



Display 1.5

Back

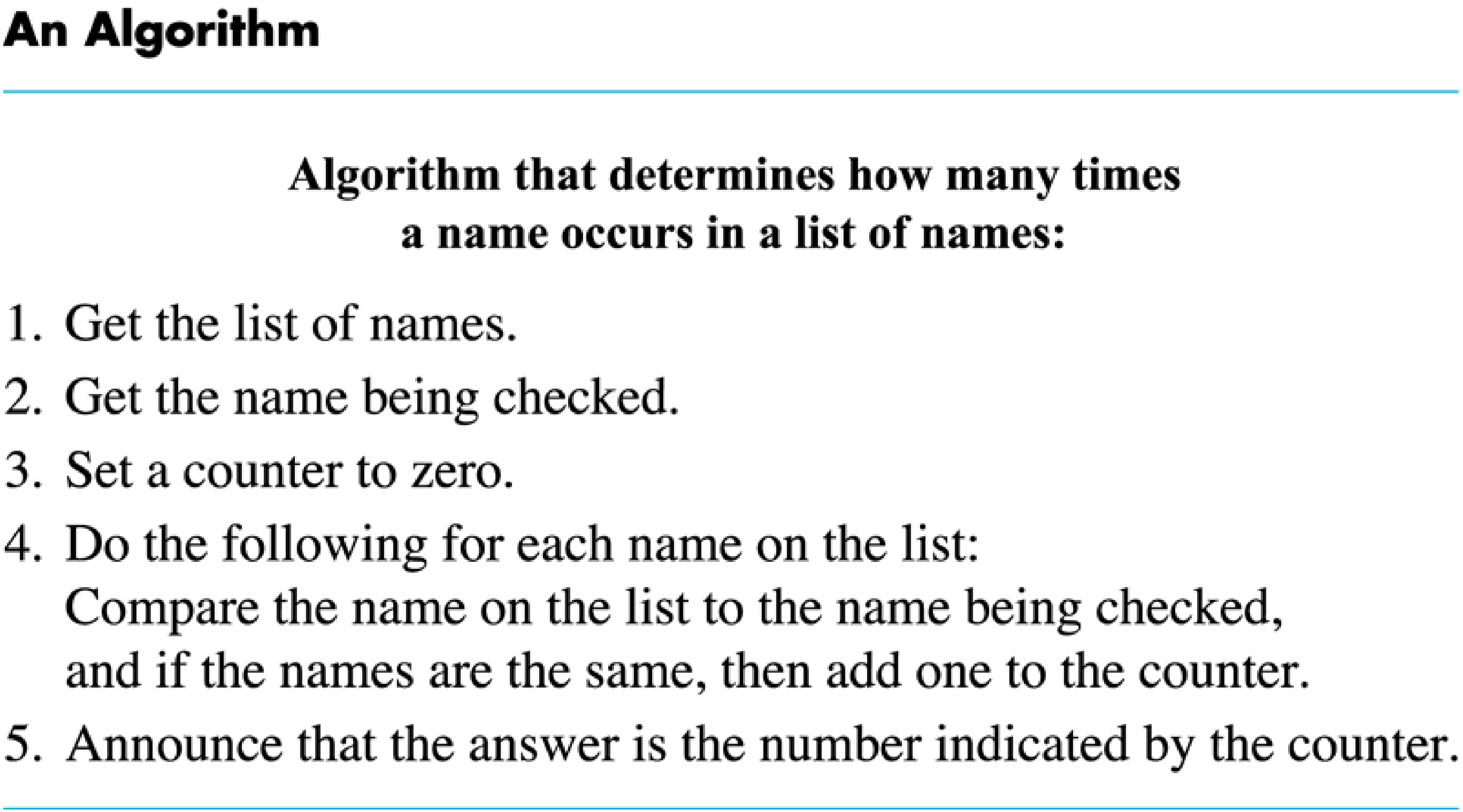
Next


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

Display 1.6

Back

Next



An Algorithm

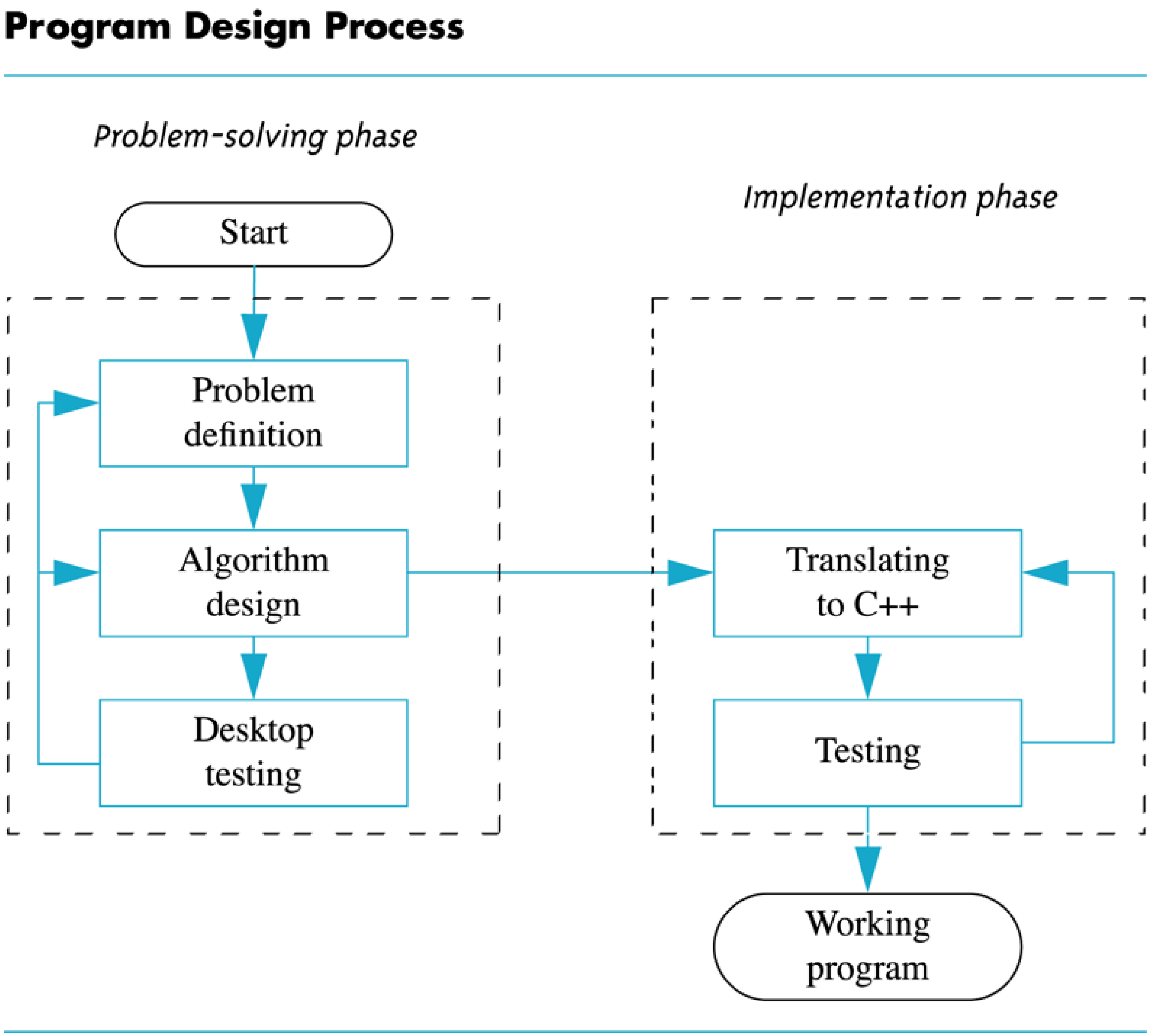
Algorithm that determines how many times a name occurs in a list of names:

1. Get the list of names.
2. Get the name being checked.
3. Set a counter to zero.
4. Do the following for each name on the list:
   Compare the name on the list to the name being checked,
   and if the names are the same, then add one to the counter.
5. Announce that the answer is the number indicated by the counter.

Display 1.7

Back

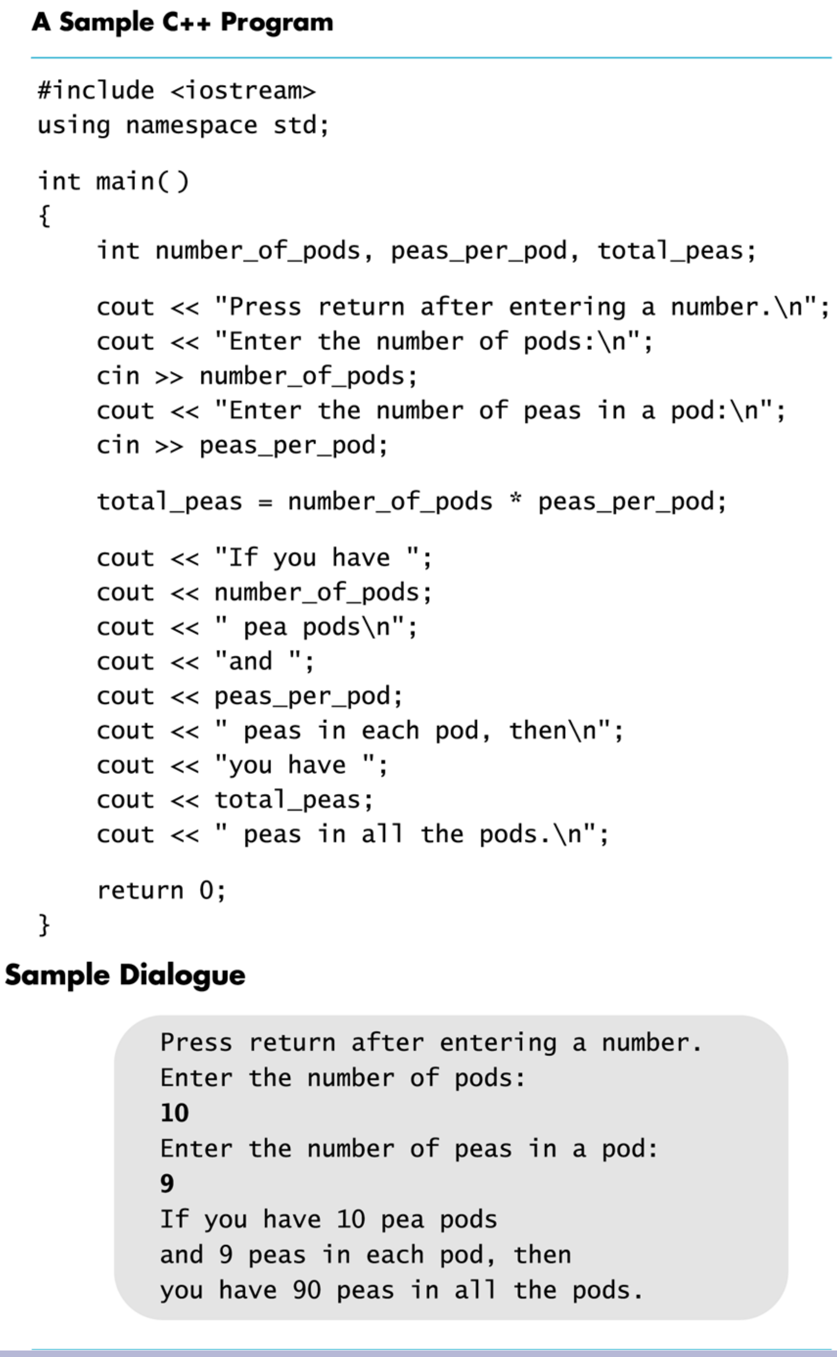
Next



Display 1.8

Back

Next



A Sample C++ Program

```
#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;
}
```

Sample Dialogue

```
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 the pods.
```