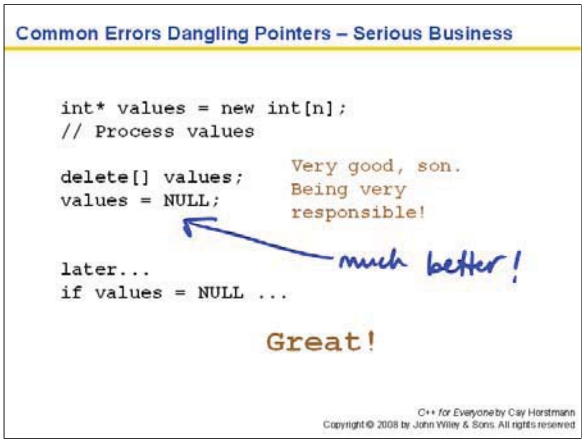


Common Errors Dangling Pointers – Serious Business

```
int* values = new int[n];
// Process values

delete[] values;
values = NULL;
```

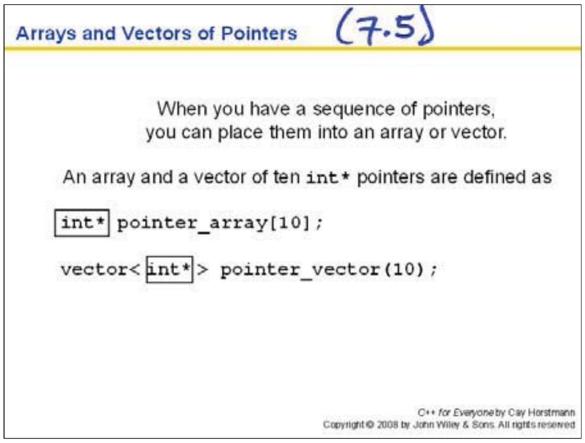
Very good, son. Being very responsible!

much better!

```
later...
if values = NULL ...
```

Great!

C++ for Everyone by Cay Horstmann
Copyright © 2008 by John Wiley & Sons. All rights reserved



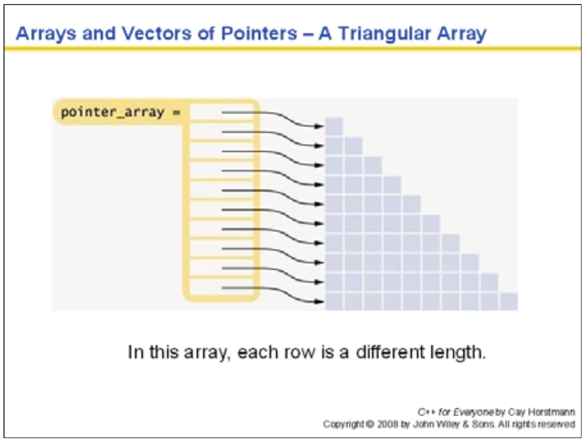
Arrays and Vectors of Pointers (7.5)

When you have a sequence of pointers, you can place them into an array or vector.

An array and a vector of ten `int*` pointers are defined as

```
int* pointer_array[10];

vector<int*> pointer_vector(10);
```

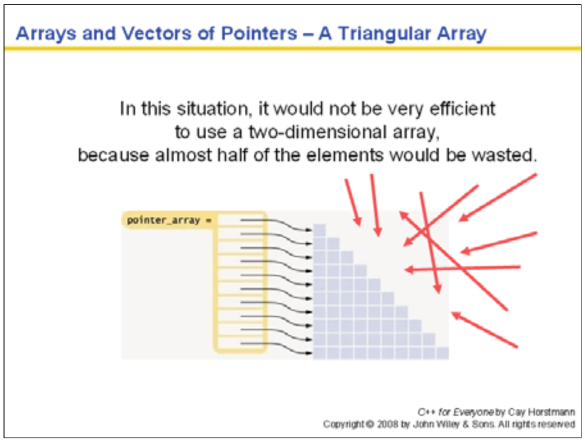
C++ for Everyone by Cay Horstmann
Copyright © 2008 by John Wiley & Sons. All rights reserved



Arrays and Vectors of Pointers – A Triangular Array

In this array, each row is a different length.

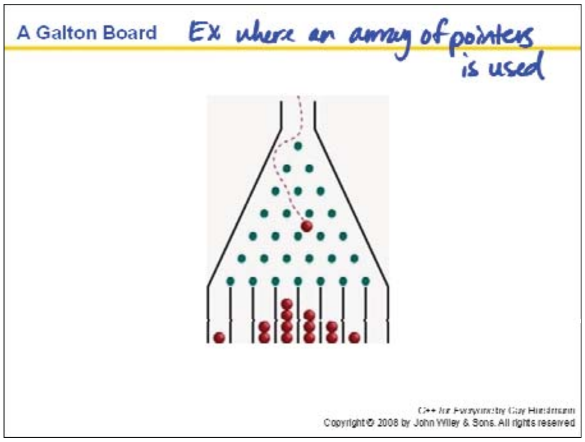
C++ for Everyone by Cay Horstmann
Copyright © 2008 by John Wiley & Sons. All rights reserved



Arrays and Vectors of Pointers – A Triangular Array

In this situation, it would not be very efficient to use a two-dimensional array, because almost half of the elements would be wasted.

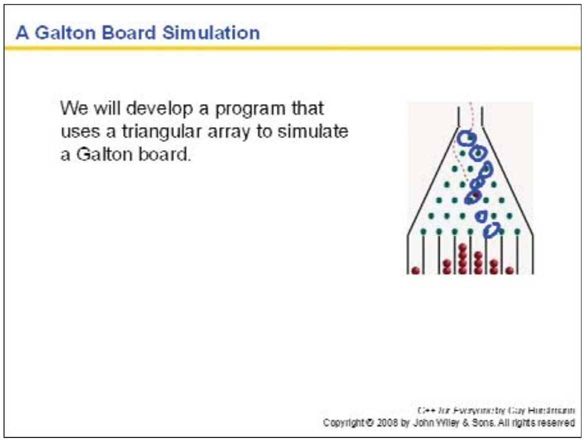
C++ for Everyone by Cay Horstmann
Copyright © 2008 by John Wiley & Sons. All rights reserved



A Galton Board

EX where an array of pointers is used

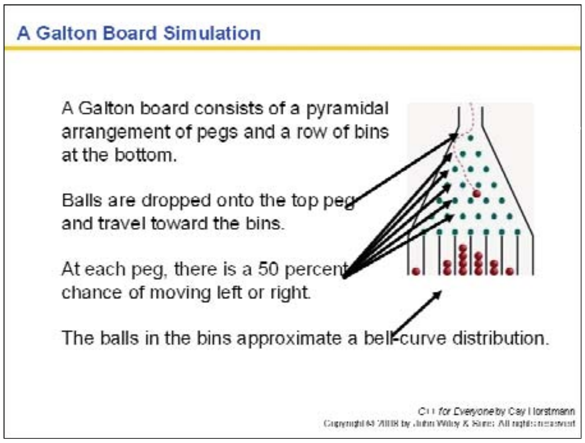
C++ for Everyone by Cay Horstmann
Copyright © 2008 by John Wiley & Sons. All rights reserved



A Galton Board Simulation

We will develop a program that uses a triangular array to simulate a Galton board.

C++ for Everyone by Cay Horstmann
Copyright © 2008 by John Wiley & Sons. All rights reserved



A Galton Board Simulation

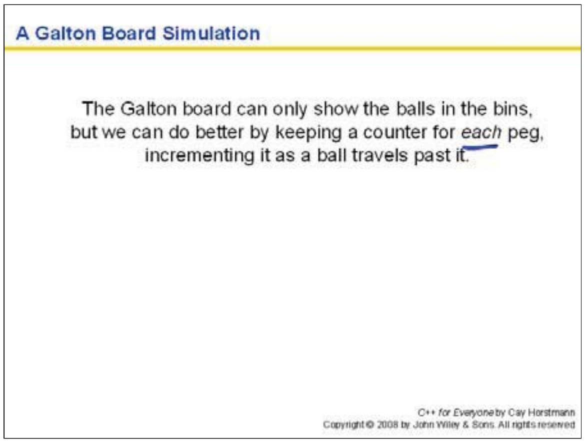
A Galton board consists of a pyramidal arrangement of pegs and a row of bins at the bottom.

Balls are dropped onto the top peg and travel toward the bins.

At each peg, there is a 50 percent chance of moving left or right.

The balls in the bins approximate a bell-curve distribution.

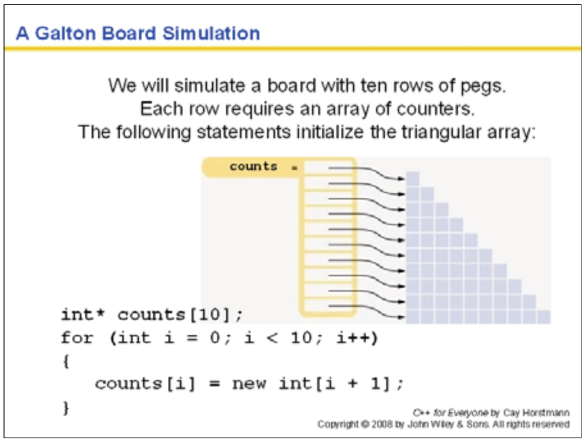
C++ for Everyone by Cay Horstmann
Copyright © 2008 by John Wiley & Sons. All rights reserved



A Galton Board Simulation

The Galton board can only show the balls in the bins, but we can do better by keeping a counter for *each* peg, incrementing it as a ball travels past it.

C++ for Everyone by Cay Horstmann
Copyright © 2008 by John Wiley & Sons. All rights reserved

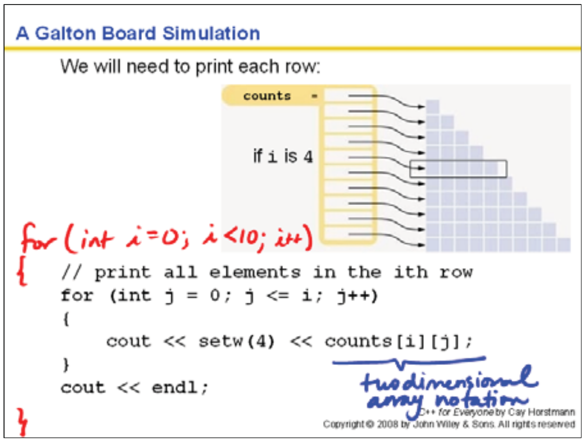


A Galton Board Simulation

We will simulate a board with ten rows of pegs.
Each row requires an array of counters.
The following statements initialize the triangular array:

```
int* counts[10];
for (int i = 0; i < 10; i++)
{
   counts[i] = new int[i + 1];
}
```

C++ for Everyone by Cay Horstmann
Copyright © 2008 by John Wiley & Sons. All rights reserved



A Galton Board Simulation

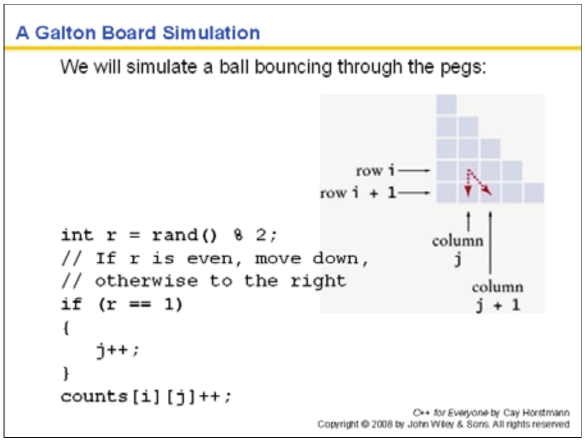
We will need to print each row:

counts =
if i is 4

```
for (int i=0; i<10; i++)
{  // print all elements in the ith row
   for (int j = 0; j <= i; j++)
   {
      cout << setw(4) << counts[i][j];
   }
   cout << endl;
}
```

two dimensional array notation

C++ for Everyone by Cay Horstmann
Copyright © 2008 by John Wiley & Sons. All rights reserved



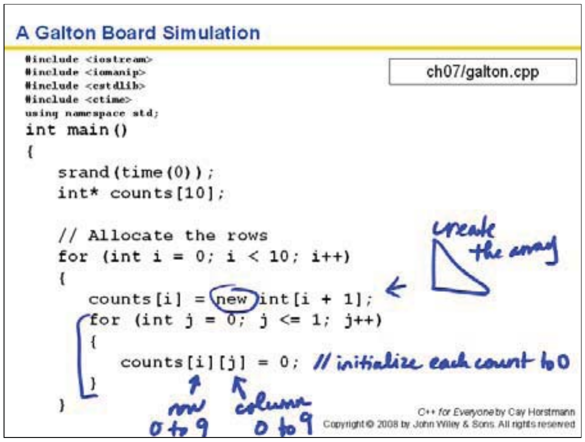
A Galton Board Simulation

We will simulate a ball bouncing through the pegs:

row i
row i + 1
column j
column j + 1

```
int r = rand() % 2;
// If r is even, move down,
// otherwise to the right
if (r == 1)
{
   j++;
}
counts[i][j]++;
```

C++ for Everyone by Cay Horstmann
Copyright © 2008 by John Wiley & Sons. All rights reserved



A Galton Board Simulation

ch07/galton.cpp

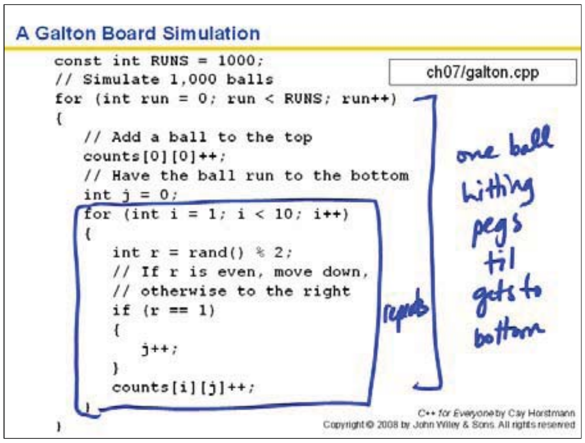
```
#include <iostream>
#include <iomanip>
#include <cstdlib>
#include <ctime>
using namespace std;
int main()
{
   srand(time(0));
   int* counts[10];

   // Allocate the rows
   for (int i = 0; i < 10; i++)
   {
      counts[i] = new int[i + 1];
      for (int j = 0; j <= 1; j++)
      {
         counts[i][j] = 0; // initialize each count to 0
      }
   }
```

create the array

row 0 to 9

column 0 to 9

C++ for Everyone by Cay Horstmann
Copyright © 2008 by John Wiley & Sons. All rights reserved



A Galton Board Simulation

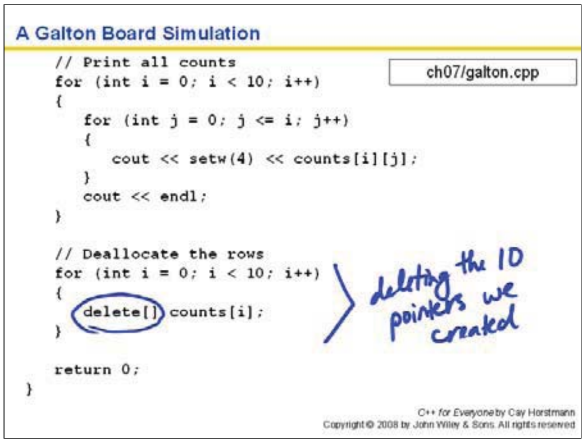
ch07/galton.cpp

```
const int RUNS = 1000;
// Simulate 1,000 balls
for (int run = 0; run < RUNS; run++)
{
   // Add a ball to the top
   counts[0][0]++;
   // Have the ball run to the bottom
   int j = 0;
   for (int i = 1; i < 10; i++)
   {
      int r = rand() % 2;
      // If r is even, move down,
      // otherwise to the right
      if (r == 1)
      {
         j++;
      }
      counts[i][j]++;
   }
}
```

repeats

one ball hitting pegs til gets to bottom

C++ for Everyone by Cay Horstmann
Copyright © 2008 by John Wiley & Sons. All rights reserved



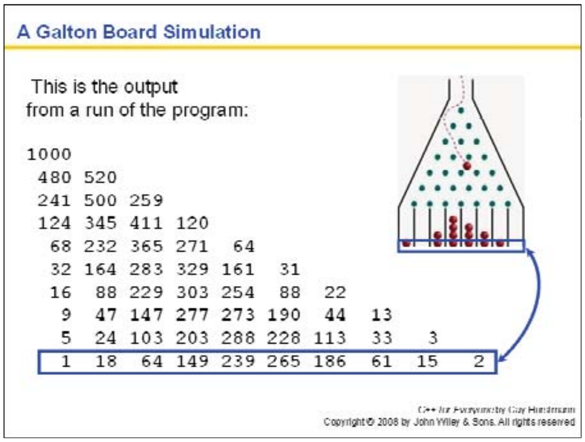
A Galton Board Simulation

ch07/galton.cpp

```
   // Print all counts
   for (int i = 0; i < 10; i++)
   {
      for (int j = 0; j <= i; j++)
      {
         cout << setw(4) << counts[i][j];
      }
      cout << endl;
   }

   // Deallocate the rows
   for (int i = 0; i < 10; i++)
   {
      delete[] counts[i];
   }

   return 0;
}
```

deleting the 1D pointers we created

C++ for Everyone by Cay Horstmann
Copyright © 2008 by John Wiley & Sons. All rights reserved

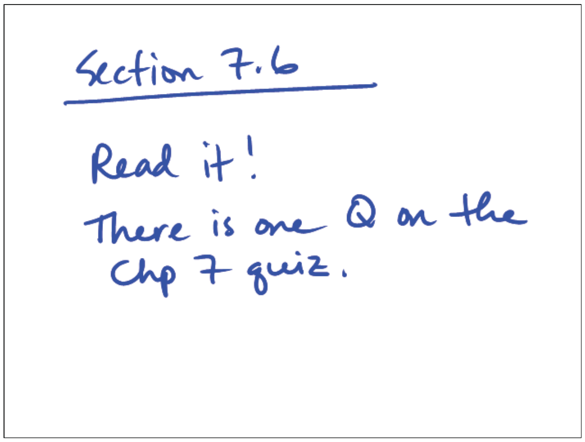


A Galton Board Simulation

This is the output
from a run of the program:

```
1000
 480 520
 241 500 259
 124 345 411 120
  68 232 365 271  64
  32 164 283 329 161  31
  16  88 229 303 254  88  22
   9  47 147 277 273 190  44  13
   5  24 103 203 288 228 113  33   3
   1  18  64 149 239 265 186  61  15   2
```

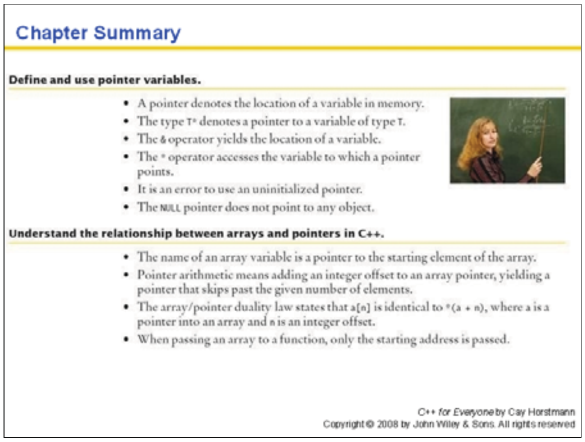
C++ for Everyone by Cay Horstmann
Copyright © 2008 by John Wiley & Sons. All rights reserved



Section 7.6

Read it!

There is one Q on the Chp 7 quiz.



Chapter Summary

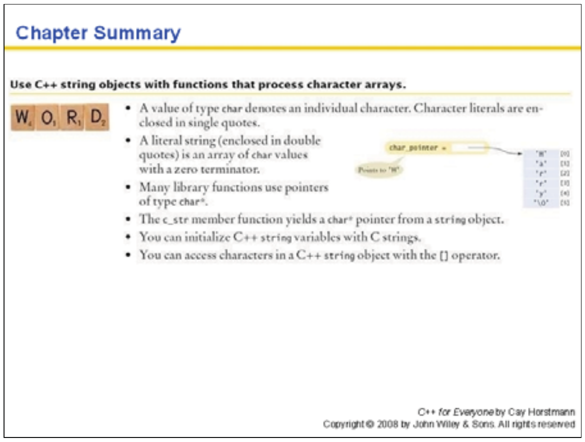
Define and use pointer variables.

- A pointer denotes the location of a variable in memory.
- The type T* denotes a pointer to a variable of type T.
- The & operator yields the location of a variable.
- The * operator accesses the variable to which a pointer points.
- It is an error to use an uninitialized pointer.
- The NULL pointer does not point to any object.

Understand the relationship between arrays and pointers in C++.

- The name of an array variable is a pointer to the starting element of the array.
- Pointer arithmetic means adding an integer offset to an array pointer, yielding a pointer that skips past the given number of elements.
- The array/pointer duality law states that a[n] is identical to *(a + n), where a is a pointer into an array and n is an integer offset.
- When passing an array to a function, only the starting address is passed.

C++ for Everyone by Cay Horstmann
Copyright © 2008 by John Wiley & Sons. All rights reserved

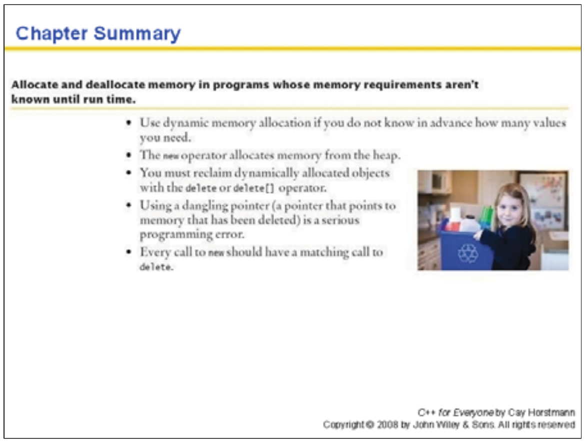


Chapter Summary

Use C++ string objects with functions that process character arrays.

- A value of type char denotes an individual character. Character literals are enclosed in single quotes.
- A literal string (enclosed in double quotes) is an array of char values with a zero terminator.
- Many library functions use pointers of type char*.
- The c_str member function yields a char* pointer from a string object.
- You can initialize C++ string variables with C strings.
- You can access characters in a C++ string object with the [] operator.

C++ for Everyone by Cay Horstmann
Copyright © 2008 by John Wiley & Sons. All rights reserved



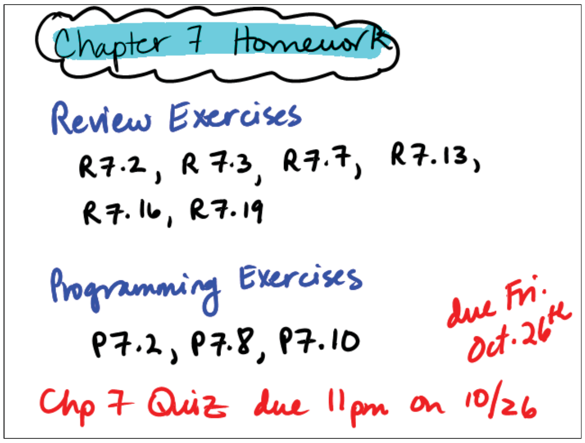
Chapter Summary

Allocate and deallocate memory in programs whose memory requirements aren't known until run time.

- Use dynamic memory allocation if you do not know in advance how many values you need.
- The `new` operator allocates memory from the heap.
- You must reclaim dynamically allocated objects with the `delete` or `delete[]` operator.
- Using a dangling pointer (a pointer that points to memory that has been deleted) is a serious programming error.
- Every call to `new` should have a matching call to `delete`.

C++ for Everyone by Cay Horstmann

Copyright © 2008 by John Wiley & Sons. All rights reserved



Chapter 7 Homework

Review Exercises

R7.2, R 7.3, R7.7, R7.13,
R7.16, R7.19

Programming Exercises

P7.2, P7.8, P7.10

due Fri.
Oct. 26th

Chp 7 Quiz due 11pm on 10/26