Topic 5

Arrays and Vectors of Pointers

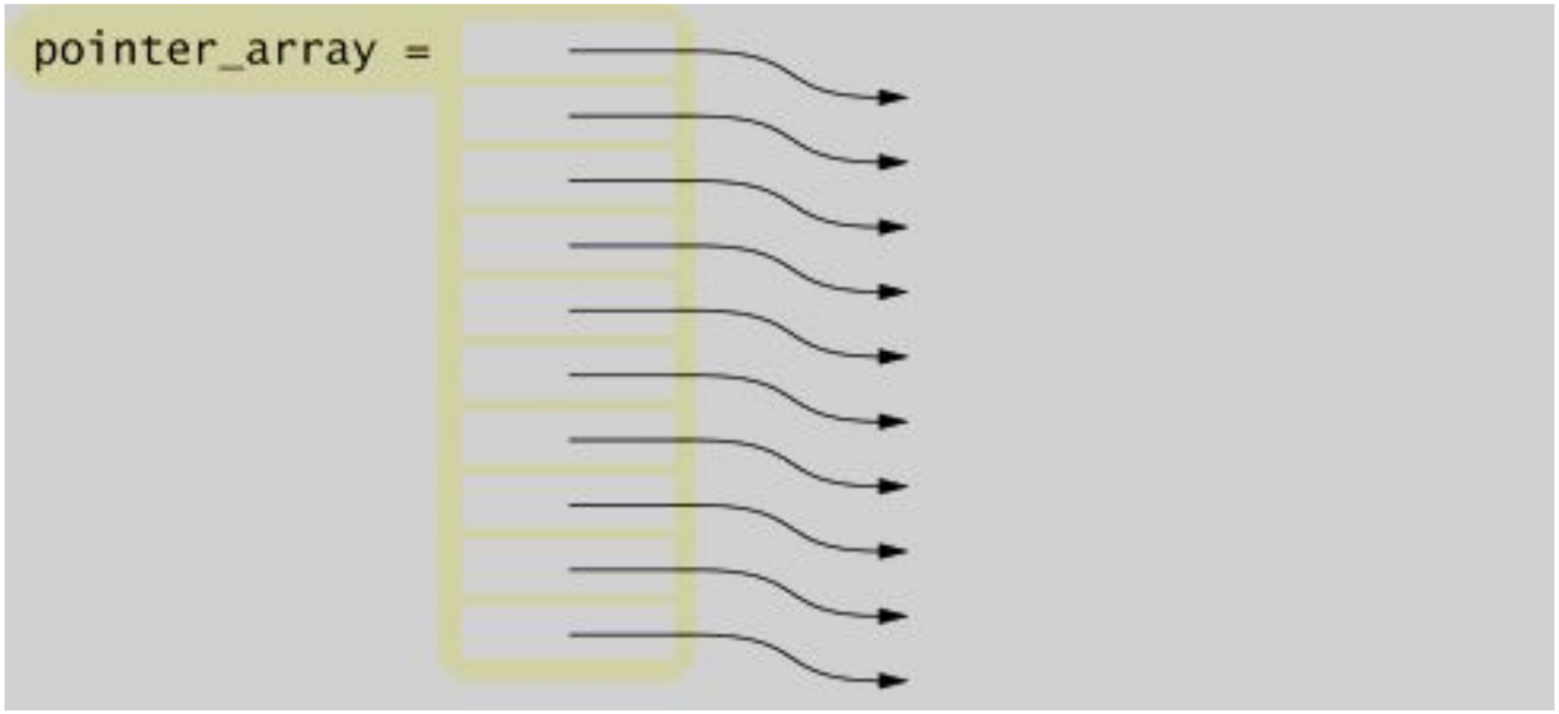
When you have a sequence of pointers,
you can place them into an array or vector.

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

```
int* pointer_array[10];

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

Arrays and Vectors of Pointers – A Triangular Array



In this array, each row is a different length. It would be inefficient to use a two-dimensional array, because almost half of the elements would be wasted

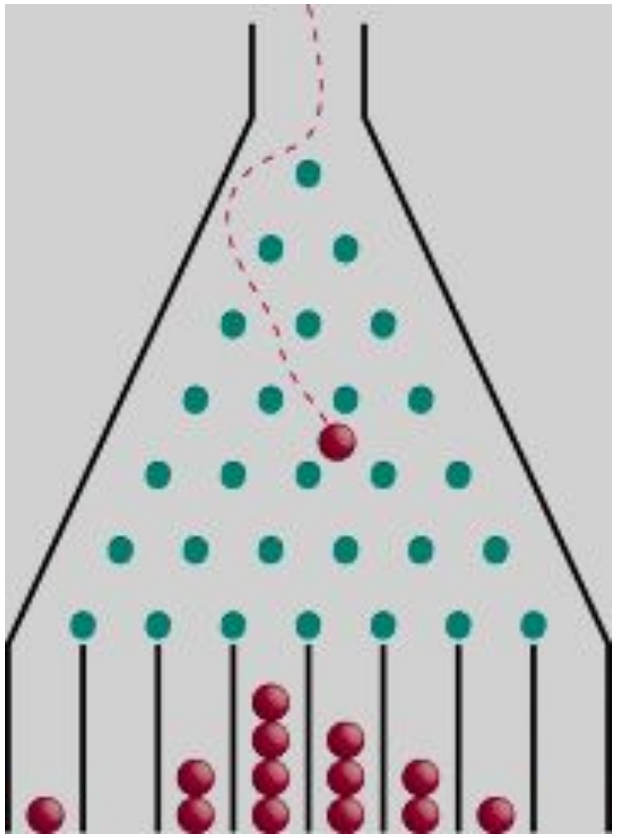
Program Example: A Galton Board

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.

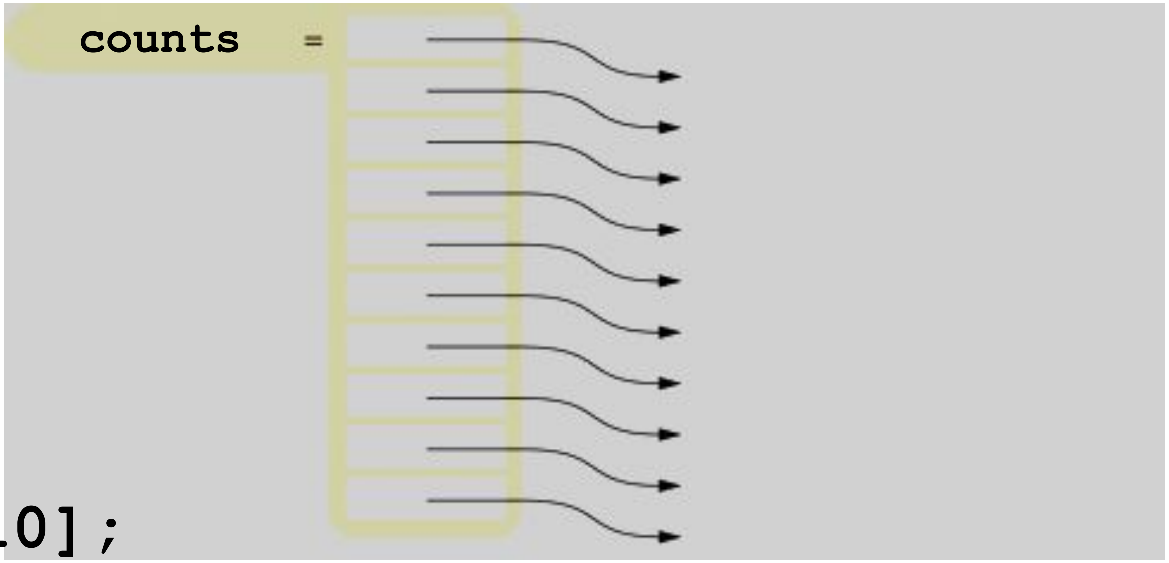
At each peg, there is a 50 percent chance of moving left or right.

The ball counts in the bins approximate a bell-curve distribution.



A Galton Board Simulation

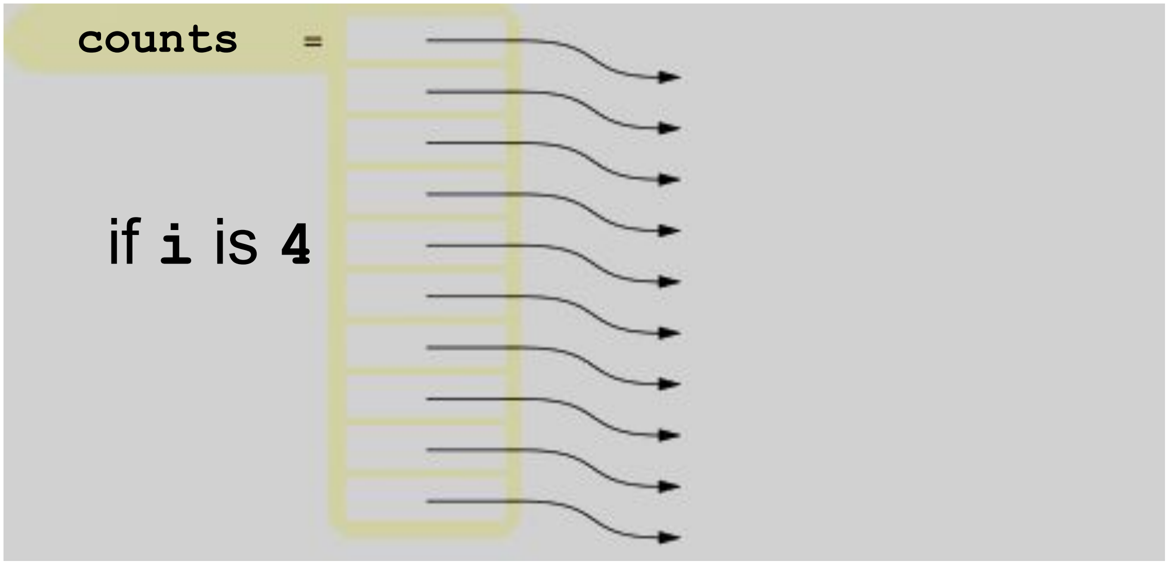
We will simulate a board with ten rows of pegs.
Each row requires an array of counters.
The following statements initialize the triangular array:



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

A Galton Board Simulation: Printing Rows

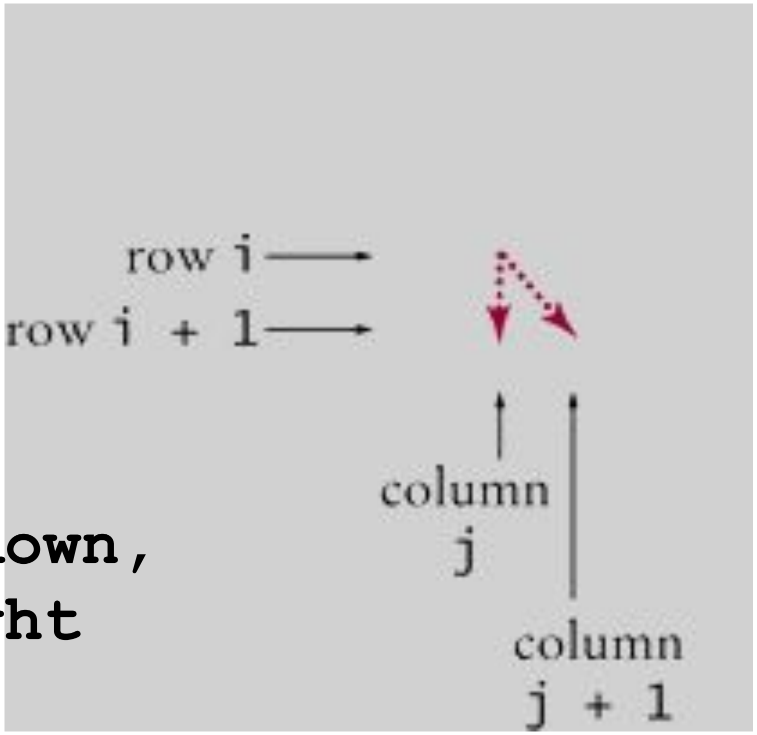
We will need to print each row:



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

A Galton Board Simulation: Ball Bouncing on Pegs

We will simulate a ball bouncing through the pegs:



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

A Galton Board Simulation: Complete Code Part 1

```
#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;
      }
   }
```

A Galton Board Simulation: Complete Code Part 2

```
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]++;
   }
}
```

A Galton Board Simulation: Complete Code Part 3

```
    // 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;
}
```

A Galton Board Simulation: Results

This is the output from a run of the program, with each number being a count of the balls that hit that peg in the triangle.

Note the bell-curve distribution of balls on the "bottom line":

```
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
```

Topic 6

Problem Solving with Pointer Pictures

- When designing programs that use pointers, you want to visualize how the pointers connect the data.

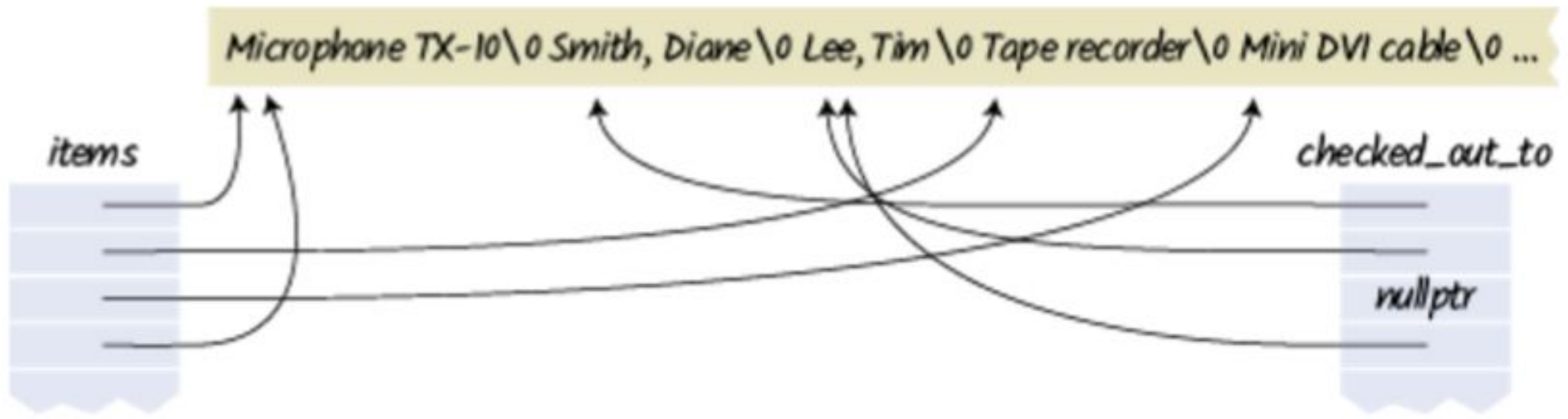
1. Draw the data blocks that will be accessed or modified through the pointers.
2. Then draw the pointer variables.
3. Finally, draw the pointers as arrows between those blocks. You may need to draw several diagrams that show how the pointers change.

Problem Solving with Pointer Pictures: Example

The media center loans out equipment (microphones, cables, and so on)

We want to track the name of each item, and the name of the user.

- All equipment and user names are in a long array of characters. New names are added to the end as needed.
- Pointers to the equipment names are stored in an array of pointers called `items`.
- A parallel array `checked_out_to` of pointers to user names. Sometimes, items can be checked out to the same user. Other items aren't checked out at all—the user name pointer is `nullptr`.


Microphone TX-10\0 Smith, Diane\0 Lee, Tim\0 Tape recorder\0 Mini DVI cable\0 ...
items
checked_out_to
nullptr

Embedded Systems

- An **embedded system** is a computer system that controls a device.
  - a processor and other hardware controlled by a computer program.
  - Unlike a personal computer, which is flexible and runs many different computer programs, the embedded system is tailored to a specific device.
  - increasingly common, in routers, washing machines, medical equipment, cell phones, automobiles, and spacecraft.
  - Probably programmed in C or C++
- Unlike PCs, embedded systems are:
  - cost sensitive: sold in quantities of millions for low prices, having little memory and slow processor
  - mission critical: most must be reliable and bug-free, as changing their program code is non-trivial and can mean life/death in the case of cars and spacecraft
  - optimized to a particular task, which may require significant code streamlining to adapt a cheap CPU to a real-time need