Dynamic Memory Allocation

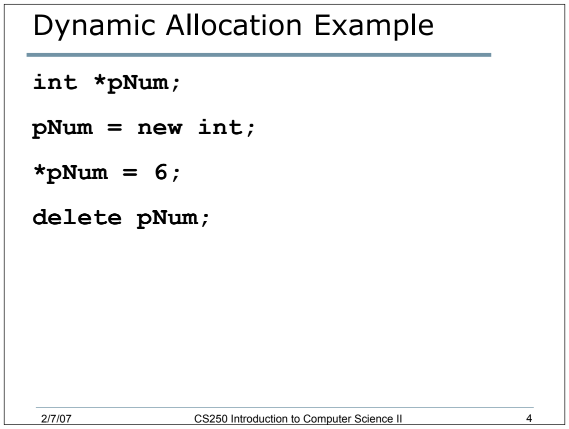
Section 10.8

Arrays

- Recall that when creating arrays, the size must be specified during compile time
- We cannot ask the user for the size of the array, then create an array of that size

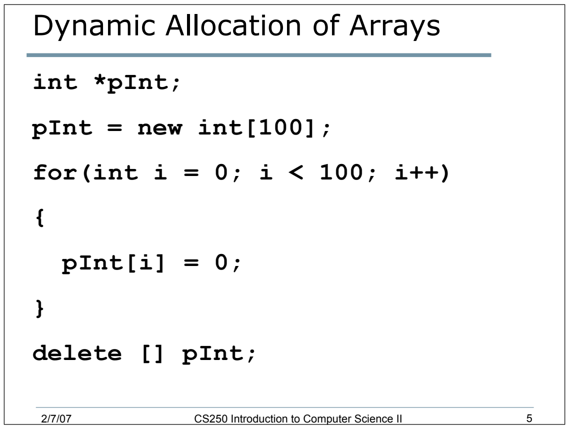
Dynamic Allocation

- It is possible to allow a program to create its own variable during run-time
- While the program is running, ask the computer to allocate enough memory to store the variable
- To do this, we use pointers and the **`new`** operator



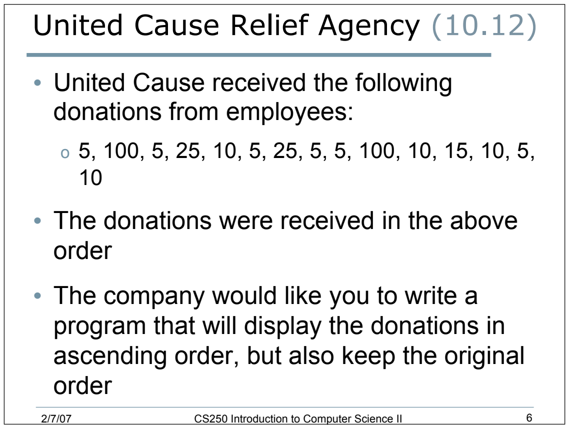
Dynamic Allocation Example

```
int *pNum;

pNum = new int;

*pNum = 6;

delete pNum;
```



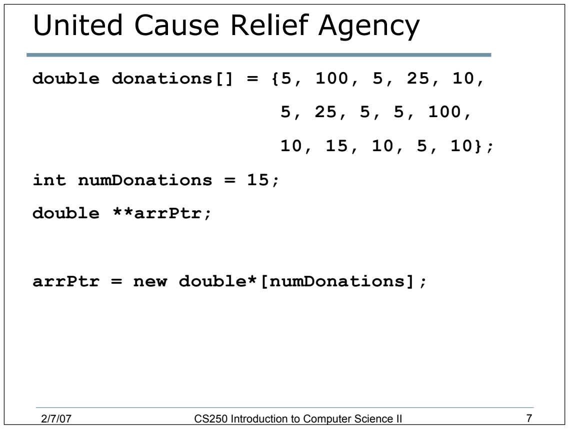
Dynamic Allocation of Arrays

```
int *pInt;

pInt = new int[100];

for(int i = 0; i < 100; i++)

{

  pInt[i] = 0;

}

delete [] pInt;
```



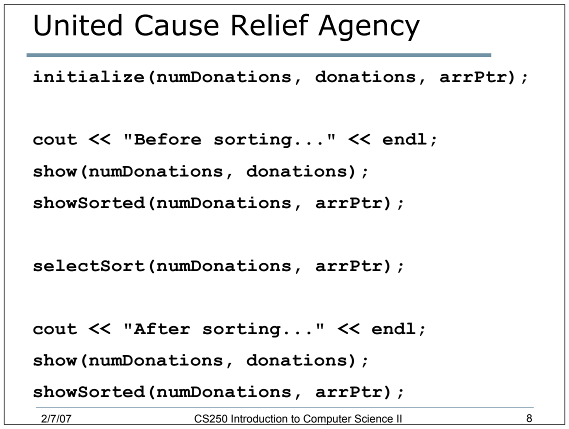
United Cause Relief Agency (10.12)

- United Cause received the following donations from employees:
  - 5, 100, 5, 25, 10, 5, 25, 5, 5, 100, 10, 15, 10, 5, 10
- The donations were received in the above order
- The company would like you to write a program that will display the donations in ascending order, but also keep the original order



United Cause Relief Agency

```
double donations[] = {5, 100, 5, 25, 10,
                      5, 25, 5, 5, 100,
                      10, 15, 10, 5, 10};
int numDonations = 15;
double **arrPtr;

arrPtr = new double*[numDonations];
```



United Cause Relief Agency

```
initialize(numDonations, donations, arrPtr);

cout << "Before sorting..." << endl;
show(numDonations, donations);
showSorted(numDonations, arrPtr);

selectSort(numDonations, arrPtr);

cout << "After sorting..." << endl;
show(numDonations, donations);
showSorted(numDonations, arrPtr);
```