Input and Output of Arrays

Lecture 29
Section 8.3

Robb T. Koether

Hampden-Sydney College

Mon, Nov 11, 2019

Outline

Reading into an Array

- We would like to read a list of numbers and store them in an array.
- The problem is that we have not seen the list yet, so we do not know how large it is.
- Yet we must reserve enough space in the array to store the list.
- What to do?

Reading into an Array

Reading into an Array

```
const int MAX_SIZE = 100;
int arr[MAX_SIZE];
```

- Use a constant integer to declare the size of the array.
- Make the value large enough to cover any reasonable case.
- Then ensure that we never read more than that many values.

Reading into an Array

- Typically, we use a loop to read values into an array.
  - On the $\texttt{i}^{\text{th}}$ iteration, the $\texttt{i}^{\text{th}}$ value is read into the $\texttt{i}^{\text{th}}$ array position.
- The loop may be controlled in any of the usual ways.
  - By a sentinel value (unknown size).
  - By EOF (unknown size).
  - By a counter (known size).
  - By a **`for`** loop (known size).

Loops Controlled by a Sentinel Value

Loop Controlled by a Sentinel Value

```
const int SENTINEL = -1;
int i = 0;
int value;
cin >> value;
while (i < MAX_SIZE && value != SENTINEL)
{
    arr[i] = value;
    i++;
    cin >> value;
}
int size = i;
```

- Loop controlled by a sentinel value.
- Be careful not to store the sentinel value in the array!

Loops Controlled by EOF

Loop Controlled by EOF

```
int i = 0;
while (i < MAX_SIZE && cin >> arr[i])
{
    i++;
}
int size = i;
```

- Loop controlled by EOF.
- This works because C++ uses “short-circuit” evaluation.

Loops Controlled by EOF

Loop Controlled by EOF

```
int i = 0;
while (cin >> arr[i] && i < MAX_SIZE)
{
    i++;
}
int size = i;
```

- This will not work.

Loops Controlled by a Counter (`while` Loop)

Loop Controlled by a Counter (`while` Loop)

```
int size;
cin >> size;
while (size > MAX_SIZE)
{
    cout << "Size is too large. Re-enter: ";
    cin >> size;
}
int i = 0;
while (i < MAX_SIZE && i < size)
{
    cin >> arr[i];
    i++;
}
```

- Loop controlled by a counter (known size).

Loops Controlled by a Counter

Loop Controlled by a Counter

```
int size;
cin >> size;
if (size > MAX_SIZE)
{
    cout << "Size is too large. "
        << "It has been reset to " << MAX_SIZE;
    size = MAX_SIZE;
}
int i = 0;
while (i < size)
{
    cin >> arr[i];
    i++;
}
```

- Or, we could make it the smaller of `size` and `MAX_SIZE`.

`for` Loops and Arrays

`for` Loops and Arrays

```
int size;
cin >> size;
for (int i = 0; i < MAX_SIZE && i < size; i++)
{
    cin >> arr[i];
}
```

- Loop controlled by a **`for`** statement.

`for` Loops and Arrays

`for` Loops and Arrays

```
int size;
cin >> size;
size = min(size, MAX_SIZE);
for (int i = 0; i < size; i++)
{
    cin >> arr[i];
}
```

- Or, we could compare `size` to `MAX_SIZE` only once, before beginning the loop.

Outline

`for` Loops and Arrays

`for` Loops and Arrays

```
for (int i = 0; i < size; i++)
{
    cout << arr[i] << endl;
}
```

- Use a `for` loop since the size of the array is known.

Writing Arrays

- How would we output the array if we wanted the elements on one line and separated by commas?

Writing Arrays

Writing Arrays

```
if (size > 0)
    cout << arr[0];
for (int i = 1; i < size; i++)
{
    cout << ", " << arr[i];
}
```

- Every element *except the first* is preceded by a comma.
- Treat the first element as a special case.
- Then output the rest in a **`for`** loop.

Outline

Examp

- Read an array and write its elements in reverse order.
- Change the array type to `Point` or `Rational`.
- Read an array and write each element's deviation from the average of the elements.

Outline

Assignment

Assignment

- Read Section 8.3.